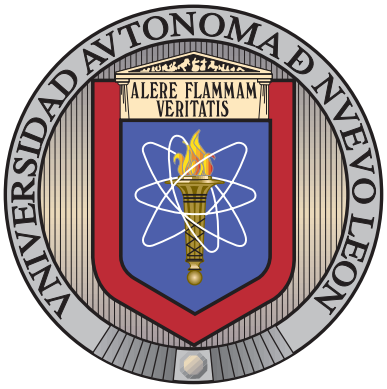
Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemáticas

Centro de Investigación en Ciencias Físico-Matemáticas



Un Problema Binivel de Localización de Almacenes Considerando Políticas de Inventarios

por

Dámaris Arizhay Dávila Soria

en opción al grado de

Doctorado en Ciencias

con Orientación en Matemáticas

San Nicolás de los Garza, Nuevo León — Febrero 9, 2022

Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemáticas

Centro de Investigación en Ciencias Físico-Matemáticas

Un Problema Binivel de Localización de Almacenes Considerando Políticas de Inventarios

por

Dámaris Arizhay Dávila Soria

en opción al grado de

Doctorado en Ciencias

con Orientación en Matemáticas

San Nicolás de los Garza, Nuevo León — Febrero 9, 2022

Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas
Centro de Investigación en Ciencias Físico-Matemáticas

Los miembros del Comité de Tesis recomendamos que la Tesis "Un Problema Binivel de Localización de Almacenes Considerando Políticas de Inventarios", realizada por la alumna Dámaris Arizhay Dávila Soria, con número de matrícula 1418698, sea aceptada para su defensa como opción al grado de Doctorado en Ciencias con Orientación en Matemáticas.

El Comité de Tesis

Dr. José Fernando Camacho Vallejo
Asesor

Dr. Leopodo Eduardo Cárdenas Barrón
Co-asesor

Dr. Samuel Moisés Nucamendi Guillén
Revisor

Dr. Omar Jorge Ibarra Rojas
Revisor

Dr. Álvaro Eduardo Cordero Franco
Revisor

Dr. Omar Jorge Ibarra Rojas
Coordinador del posgrado en Ciencias con Orientación en Matemáticas

San Nicolás de los Garza, Nuevo León, Febrero 9, 2022

Dedicatoria

A las personas mas importantes en mi vida:
Daniel Dávila Ramírez,
María Luisa Soria,
y Juan Daniel Dávila Soria.

Y a esos seres de cuatro patitas que me acompañaron en este viaje:
Pelusa (†) y Pancho.

Agradecimientos

Quiero comenzar agradeciendo a mi familia, por ser mi soporte en todo momento, por proporcionarme las bases necesarias para ser la persona que ahora soy y por apoyarme incondicionalmente en mis decisiones.

También agradezco a mis profesores, por compartirme sus conocimientos y ayudarme a crecer en lo profesional.

Al Dr. Fernando Camacho, por aceptar ser mi asesor desde Maestría hasta Doctorado, por guiarme y apoyarme en mi trabajo de tesis y ayudarme siempre que lo necesité.

A mi coasesor el Dr. Leopoldo Cárdenas y al comité de tesis: Dr. Samuel Nucamendi, Dr. Omar Ibarra y Dr. Álvaro Cordero por ayudarme a mejorar mi trabajo con todas sus recomendaciones y sugerencias.

A mis compañeros de posgrado por tantas risas y momentos compartidos.

Y en general, a todas aquellas personas que de manera directa o indirecta hayan contribuído a la culminación de este trabajo de tesis durante los últimos años.

Resumen

En esa tesis se presenta un problema de localización y asignación que ocurre en una compañía encargada de ubicar almacenes para suministrar productos a un conjunto de clientes. La compañía tiene el objetivo de minimizar el costo total de localizar los almacenes y el costo relacionado con la cantidad del pedido hecho en cada uno de ellos. La cantidad del pedido que se realiza en los almacenes es una decisión hecha de forma independiente a la decisión de localizarlos. Es decir, una vez que la compañía toma la decisión de donde establecer los almacenes, el tomador de decisiones asociado con cada uno de los almacenes debe de realizar su propia política de inventario para determinar la cantidad de pedido a realizar. Los almacenes tienen permitido tener cierto número máximo de ordenes atrasadas, las cuales representan un costo extra para ellos. Esta situación puede ser modelada como un problema binivel, donde el nivel superior está asociado con la compañía que necesita minimizar los costos relacionados con la localización y asignación y el costo total de la orden realizada en cada almacén. Mientras que cada almacén es asociado con un nivel inferior, los cuales son independientes entre si, en donde se quiere minimizar el costo total de inventario. El problema binivel resultante es un problema con una función objetivo en el nivel superior y múltiples problemas no lineales en cada nivel inferior, lo cual es difícil de resolver a optimalidad. Para resolver este problema se propone un algoritmo metaheurístico de lluvia de ideas debido a su reciente uso para problemas binivel. Para resolver a optimalidad cada problema asociado con el nivel inferior, y tomando en cuenta la no linealidad existente, se utiliza el método de multiplicadores de Lagrange. Con esto se logra tener soluciones binivel factibles. Para medir el comportamiento del algoritmo, se consideraron tres tipos de instancias diferentes, y los resultados obtenidos fueron comparados con los resultados obtenidos con un algoritmo exhaustivo (cuando fue posible resolverse) y con un algoritmo genético con la finalidad de tener otro metaheurístico con el cual comparar. Los resultados muestran que la optimalidad es alcanzada por el algoritmo de lluvia de ideas para las instancias pequeñas y medianas, mientras que para las grandes, donde el algoritmo exhaustivo no es capaz de encontrar el óptimo en un tiempo límite, el algoritmo de lluvia de ideas alcanza los mismos o mejores resultados en promedio que el

algoritmo genético. Finalmente se presentan algunas conclusiones sobre la investigación aquí realizada y posibles extensiones de trabajos a futuro.

Contribuciones

Los problemas de localización han sido estudiados durante largo tiempo, tomando cada vez mayor auge debido a sus diversas aplicaciones. En especial, los problemas de localización modelados mediante programación binivel pueden observarse en la literatura en conjunto con otras decisiones de la cadena de suministros. Aun así, debido a los nuevos retos de la industria, aparecen áreas de oportunidad para generar aportes en el área de programación binivel. Motivados por lo anterior, se ha realizado esta investigación doctoral.

La siguiente lista presenta las contribuciones principales del presente trabajo de tesis realizado durante los tres años que comprende el programa.

- Se abordó un problema de localización modelado como un problema de programación binivel. Ya que al realizar la revisión de literatura se observó que no hay algún trabajo binivel que considere en conjunto las decisiones de localización y asignación.
- Se definen características especiales del modelo, las cuales son explotadas por el algoritmo propuesto.
- Se propone un algoritmo metaheurístico de lluvia de ideas para resolver el problema propuesto. Esto debido al hecho de dicho metaheurístico ha sido utilizado recientemente de manera exitosa para resolver problemas de programación binivel.
- Se adaptaron instancias de la literatura de problemas de localización en conjunto con decisiones de inventario para tenerlas como base para futuros problemas modelados de este tipo.
- Se realizaron pruebas con las instancias generadas y los resultados se compararon contra un algoritmo exhaustivo y un algoritmo genético, mostrando la estabilidad y robustez del algoritmo de lluvia de ideas propuesto en esta tesis.
- Por último, se plantean diversas formas de extender el trabajo a modelos futuros que involucren decisiones con demanda estocástica.

Índice general

Capítulo 1

Introducción

Los problemas de la cadena de suministro son considerados un área importante de estudio dentro de la investigación de operaciones, y es por esto que se ha dedicado mucho énfasis al estudio de este tipo de problemas durante las últimas décadas.

Una cadena de suministros se constituye por diferentes actores de entre los cuales destacan los proveedores, centros de producción, almacenes, centros de distribución y clientes. Hay que tomar en cuenta que cada uno de estos actores tiene diferentes tipo de de decisiones que llevar a cabo.

Une ejemplo de estas decisiones puede ser la de donde localizar ciertos almacenes. En este caso, las empresas deben considerar diversos factores al momento de decidir donde abrirán sus almacenes, tales factores pueden ser: la cercanía con las empresas o los clientes a los cuales servirán, la satistacción de sus clientes, los tiempos de reordenamiento, los tipos de productos que se manejarán, entre otros. Debido a esto, es considerada una decisión de gran importancia ya que de esto puede depender el éxito o fracaso de toda la cadena.

Usualmente, después de que se realiza el proceso de localización y asignación existen otras decisiones que deben ser tomadas por las empresas, tales como el ruteo o políticas de inventarios. En la parte de ruteo se asigna el orden en que los clientes que serán abastecidos desde cada almacén deben ser visitados. Mientras que en la parte de inventarios se toma la decisión de la cantidad del pedido a realizar en cada instalación dependiendo de la demanda requerida por los clientes.

Teniendo en cuenta lo anterior, en este trabajo de tesis doctoral se plantea un problema en el que se incorporan las decisiones de localización e inventarios dentro del mismo problema. Además, se considera una jerarquía entre ambas decisiones y algunas consideraciones específicas sobre las cantidades de pedido. Un planteamiento más detallado se presenta en la siguiente sección.

1.1 Descripción del problema

En este trabajo consideramos una situación en la que una empresa requiere abrir cierto número de almacenes para abastecer productos a clientes, dicho clientes mantienen una demanda fija de pedido independientemente de cuales almacenes están abiertos. El objetivo de la empresa es minimizar los costos de localizar sus almacenes y asignar los clientes a ellos, además de los costos de los pedidos a realizar según la demanda total de cada almacén dentro de cada periodo de abastecimiento. En este caso los almacenes tienen permitido contar con un cierto número máximo de faltantes, el cual incurre en un costo extra para el almacén. Cada pedido que se realice para un almacén es tomado en cuenta dentro de los costos que la empresa minimiza al momento de realizar la asignación.

Tomando en cuenta la jerarquía presente en las decisiones a tomar en este problema, éste puede modelarse mediante un enfoque de programación binivel. En el capítulo 2 se explicará más a detalle.

1.2 Motivación

Como se mencionó anteriormente, los procesos de localización y asignación en conjunto con otras decisiones de la cadena de suministros pueden encontrarse en la literatura. En especial, la decisión de localización y asignación junto con la de inventarios puede verse en trabajos como Cabrera et al. (2013) , donde se consideran dos escalones de la cadena de suministros en un mismo problema. Este trabajo ha motivado el estudio de ambas decisiones en un mismo problema, pero surge la pregunta de qué pasaría si cada almacén pudiera realizar su propia política de inventarios de forma libre y la empresa considerara la cantidad de pedido realizada por cada uno de ellos para su decisión de localización.

Una aplicación de este problema puede verse en la industria automotriz, donde empresas extranjeras que buscan posicionarse en un nuevo mercado buscan localizar sus plantas de ensamble (las cuales pueden verse como los almacenes) de manera que puedan ser abastecidas todas sus agencias (clientes). Una vez que son abiertas sus plantas, éstas a su vez tienen la necesidad de colocar órdenes de pedido de piezas dependiendo de la demanda de autos que se tenga en las agencias. Estas cantidades de pedido son de suma importancia para la compañía que abre las plantas ya que los costos de las piezas que una planta de manufactura requiera también incurre en costo de acarreo. En este caso, se consideran dos tomadores de decisiones; uno es la empresa que requiere abrir sus plantas, mientras que por otro lado, cada una de las agencias requiere realizar independientemente

sus pedidos dependiendo de la demanda de los clientes asignados a ella. Para la empresa, lo importante es minimizar los costos de localización y asignación de agencias a plantas, además de los costos de acarreo que se incurren debido a las piezas demandadas en cada una de las agencias.

Como se mencionó anteriormente, el problema propuesto se acopla de manera natural a un problema de programación binivel. Para este caso, la función objetivo del nivel superior modela la forma en que se localizan los almacenes. Mientras que el nivel inferior se encarga de la cantidad de pedido a realizar en cada uno de los almacenes.

1.3 Objetivo

Los objetivos principales de esta tesis son los siguientes:

- Elaboración de un modelo matemático que permita describir una primera aproximación a la toma de decisiones del problema propuesto.
- Proponer un algoritmo metaheurístico para la solución del modelo binivel, el cual explote las características del modelo.
- Revisar la robustez del algoritmo propuesto mediante la comparación contra los resultados de un algoritmo exhaustivo y un algoritmo genético.

1.4 Metodología

La metodología que se realizó durante esta investigación fue la siguiente:

1. Revisión de literatura sobre problemas de localización y asignación.
2. Revisión de literatura sobre problemas de inventarios.
3. Análisis y planteamiento del problema a estudiar.
4. Modelación del problema binivel
5. Análisis de métodos para resolver el problema propuesto.
6. Programación del método de Multiplicadores de Lagrange para la resolución del nivel inferior.

7. Creación de método constructivo para generar la población inicial de soluciones en el nivel superior y análisis de movimientos de combinación y permutación para el algoritmo.
8. Programación del algoritmo de lluvia de ideas para el problema binivel.
9. Creación de instancias de prueba.
10. Afinación de los parámetros involucrados en el algoritmo de lluvia de ideas.
11. Presentación de avances el la Escuela Latino-Americana de Investigación de Operaciones.
12. Obtención de resultados e implementación de algoritmo exhaustivo para tener comparación en funciones objetivo para algunas instancias.
13. Análisis de resultados y conclusiones.
14. Escritura de un artículo para publicación en revista indexada.
15. Escritura y revisión de tesis.

1.5 Estructura de la tesis

Para finalizar este primer capítulo, se describe la forma en que está estructurado este documento.

En el Capítulo 1 se dió una breve introducción y descripción sobre el problema estudiado, también se presentó la motivación para llevar a cabo esta investigación, los objetivos a seguir y la metodología seguida.

En el Capítulo 2, se presenta la revisión de literatura de algunos trabajos dedicados a los problemas de localización y asignación, para ver los diversos enfoques que se le ha dado a este tipo de problemas. Es por esto, que se presenta una descripción de las características principales de dichos problemas. Luego, se presenta una revisión de la literatura de los problemas de inventarios. Recuerde que el problema del nivel inferior de nuestro modelo propuesto es un problema de inventarios. Una sección también es dedicada a problemas de localización considerando decisiones de inventarios. Por último, se presenta la estructura general de los modelos de programación binivel y se mencionan algunas de sus

interesantes aplicaciones.

Después, el Capítulo 3 muestra el planteamiento del problema, así como el modelo matemático que se propone para resolverlo. Se define formalmente cada una de las variables, restricciones y parámetros involucrados en dicho modelo. Se concluye el capítulo discutiendo algunas características importantes del modelo.

En el Capítulo 4, se detalla el algoritmo propuesto para resolver el problema planteado en el capítulo anterior. Se hace una breve revisión de literatura de los problemas resueltos con el algoritmo de lluvia de ideas y se describe el algoritmo propuesto a detalle. Para esto, se incluyen pseudo-códigos que ilustran los componentes esenciales del algoritmo.

Luego, el Capítulo 5 presenta todo lo referente a la experimentación computacional. Esto es, se describe el ambiente computacional, la elaboración de las instancias de prueba, la afinación del algoritmo de lluvia de ideas mediante un diseño de experimentos, incluyendo algunas gráficas de efectos principales, para luego presentar los resultados y la comparación de los resultados contra un algoritmo exhaustivo (para instancias de tamaño limitado) y un algoritmo genético.

Después, en el Capítulo 6 se encuentran las conclusiones del trabajo, y se presenta una discusión y recomendaciones sobre los resultados que se obtuvieron utilizando el algoritmo propuesto en las instancias de prueba. También se enlistan algunas posibles direcciones para trabajo futuro.

Finalmente, la tesis concluye con una sección de *Anexos* donde se podrán encontrar las tablas con los datos de las instancias de prueba, así como algunos resultados computacionales.

Capítulo 2

Revisión de literatura

2.1 Problemas de localización

Los problemas de localización y asignación son una parte importante del proceso de distribución en la cadena de suministro y han sido estudiados frecuentemente durante las últimas décadas. En este tipo de problemas se cuenta con un conjunto de sitios potenciales para ubicar instalaciones que deben atender a un grupo de clientes. El objetivo es seleccionar los sitios que se abrirán, para después de eso, asignar los clientes a las instalaciones abiertas usando criterios predefinidos como puede verse en Cooper (1963).
El modelo básico de un problema de localización y asignación sin considerar capacidades puede verse de la siguiente forma:

$$\min_{x,y} \sum_{i \in I} \left[F_i x_i + \sum_{j \in J} c_{ij} y_{ij} \right] \tag{2.1}$$

s.a.:

$$\sum_{i \in I} y_{ij} = 1, \quad j \in J \tag{2.2}$$

$$y_{ij} \leq x_i, \quad i \in I, j \in J \tag{2.3}$$

$$x_i, y_{ij} \in \{0, 1\}, \quad i \in I, j \in J \tag{2.4}$$

Donde I y J corresponden al conjunto de instalaciones y clientes, respectivamente. Las variables de decisión x_i corresponden a la apertura de las instalaciones y las variables y_{ij} son relacionadas con la asignación de los clientes a las instalaciones. La función objetivo (2.1) consiste en minimizar los costos de abrir las instalaciones (F_i) y asignar los clientes a ellas (c_{ij}). La restricción (2.2) garantiza que cada cliente sea atendido por una instalación, mientras que la restricción (2.3) asegura que los clientes sean asignados solo a instalaciones que hayan sido abiertas. Y finalmente la restricción (2.4) especifica que las variables de decisión son binarias. La siguiente figura representa de manera gráfica el funcionamiento de un modelo básico de localización de instalaciones. En dicha figura se puede observar que

se tienen 17 sitios potenciales para ubicar las instalaciones (triángulos), de los cuales 4 son seleccionados para abrir instalaciones (triángulos grises). Cada una de las instalaciones abiertas tiene asignada un conjunto de clientes (círculos negros). Esa solución conlleva a un costo, y lo que se busca es encontrar la configuración de instalaciones abiertas y asignación de clientes con costo mínimo.

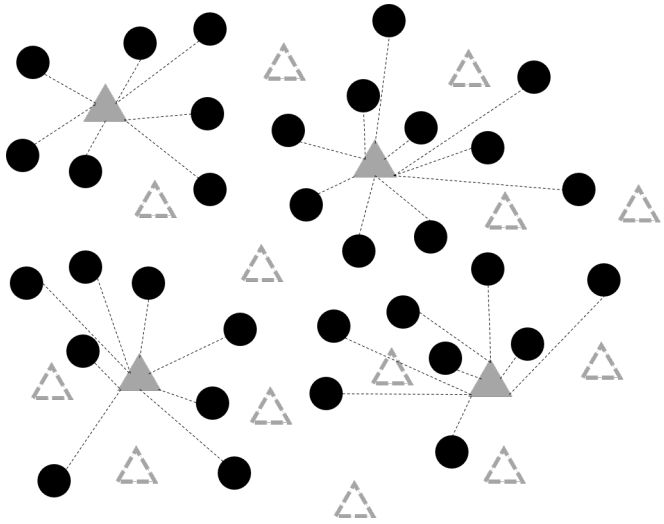


Figura 2.1: Ejemplo ilustrativo de un problema básico de localización-asignación

El modelo anterior ha sido muy estudiado en la literatura y se han considerado diversas variantes a través del tiempo. Como primer ejemplo, se puede mencionar a los problemas basados en el principio de cobertura, como lo son los del conjunto de cobertura, máxima cobertura y p-centros. En los problemas del conjunto de cobertura el objetivo es minimizar el número de instalaciones que se necesitan para cubrir todas las demandas de los clientes (ver Wu et al., 2021; Wang, Pan, Al-Shihabi, Zhou, Yang and Yin, 2021; Lien et al., 2021). En los problemas de máxima cobertura el objetivo es maximizar la demanda cubierta al abrir p instalaciones (ver Diabat and Theodorou, 2015; Jia et al., 2007; Mahmud and Indriasari, 2009), y en los modelos de p centros se busca minimizar la máxima distancia entre los clientes y las p instalaciones (ver Suzuki and Drezner, 1996; Du et al., 2020; Contardo et al., 2019). Otra variante muy interesante son los problemas basados en medianas (ver Hanjoul and Peeters, 1987; Daskin, 2008; Rahman and Smith, 2000), donde el objetivo es minimizar la distancia promedio entre los clientes y los p sitios más cercanos. Los modelos de carga fija (ver Calvete et al., 2018; Pop et al., 2017; Boonmee et al., 2017), tienen como objetivo minimizar el costo de la localización de las instalaciones y el de transporte. Por otra parte, en los modelos de la p-dispersión se busca maximizar la mínima distancia entre cualquier par de instalaciones (ver Tutunchi and Fathi, 2019; Sayah and Irnich, 2017; Ahmadi-Javid et al., 2017).

Por otro lado, de acuerdo a cuántos procesos de una cadena de suministros se involucren en el problema, los problemas de localización también se pueden clasificar en modelos de un solo escalón (ver Murray and Gerrard, 1997; Bautista and Pereira, 2006) o de múltiples escalones (ver Shen et al., 2003).

Gracias a la gran variedad de modelos de localización existentes, es posible aplicarlos para diversos enfoques. Algunos de los enfoques más comunes de este tipo de modelos son por ejemplo, al sector salud (ver Farahani et al., 2014; Chaiwuttisak et al., 2016; Gebicki et al., 2014; Şahin et al., 2007), servicios públicos (ver Schuijbroek et al., 2017; Boyacı et al., 2015; Lin and Yang, 2011; Usman et al., 2013), problemas de cadena de suministros (ver Randhawa and West, 1995; Perl and Sirisoponsilp, 1988; Acharya et al., 2006; Govindan et al., 2014; Masudin, 2013), sistemas de distribución (ver Nozick and Turnquist, 1998; Shen and Qi, 2007; Masudin, 2015; Wati and Nuha, 2018; Kusmindarti, 2009) o logística reversa (ver Diabat et al., 2015; Diabat and Theodorou, 2015; Temur et al., 2014; Lu and Bostel, 2007). Cada uno de los enfoques anteriores tienen sus propios objetivos y forma de modelar relacionado con el problema que presentan.

Una parte importante a estudiar sobre los problemas de la cadena de suministros son los de la ubicación de los almacenes, los cuales dan servicio a clientes que tienen una demanda y necesidades específicas. Es por esto que es importante dedicar una sección a este tipo de problemas.

2.2 Problema de localización de almacenes y su importancia

Determinar la ubicación de un almacén es uno de los aspectos primordiales que hay que considerar antes de su instalación. La localización del almacén es una decisión de tipo estratégico, y puede ser considerada decisiva en el éxito o el fracaso de un proceso. El estudio de la ubicación de un almacén es más complejo cuando el tamaño de la empresa y su sistema de distribución es mayor. En cualquier caso, dicho estudio debe realizarse mediante el análisis de cuatro factores básicos: la producción, los costos, la demanda y la competencia.

A continuación se presenta una revisión de algunos trabajos encontrados en la literatura de localización de almacenes con sus respectivos objetivos de estudio.

El trabajo más antiguo encontrado fue el de Baumol and Wolfe (1958), donde se aborda un método para determinar un patrón de ubicación geográfica más rentable para los almacenes que son empleados por una empresa para entregar cantidades conocidas de su producto terminado a sus clientes, donde también se permite que varíe el número de almacenes. A partir de ahí diversos enfoques han aparecido en la literatura.

Por ejemplo, problemas que consideran almacenes con capacidades pueden verse en Ozsen et al. (2008), donde se presenta un modelo de ubicación de un almacén central que tiene cierta capacidad máxima para recibir productos, en el cual se captura la interdependencia entre los problemas de capacidades y la gestión de inventario en los almacenes. El modelo toma en cuenta un sistema logístico en el que una sola planta envía un tipo de producto a un conjunto de clientes, cada uno con una demanda incierta. Otro problema de este tipo puede verse en Aghezzaf (2005) donde primero se propone un modelo determinista para el problema de localización de almacenes con capacidades, es decir, todos los parámetros relevantes se conocen con certeza. Luego se presenta un modelo de optimización para el problema cuando la demanda es incierta.

Otro tipo de enfoque de estos problemas puede verse en Zhu et al. (2021), donde se aborda una aplicación del problema de localización de almacenes cuando se tiene un conjunto de clientes a los cuales hay que dar servicio pero éstos tienen que ser surtidos por diferentes negocios. Entonces lo que se busca es tener almacenes que tengan variedad de productos para poder atender desde un solo almacén todos los productos necesarios que requiera un cliente. El modelo resultante es un modelo de programación entera. Para este problema se propone un algoritmo de agrupamiento de k enlaces para optimizar la asignación de productos entre varios almacenes basados en la distribución de pedidos de varios artículos para minimizar el número total de divisiones de pedidos.

Algunas decisiones de localización presentan cierta jerarquía en la toma de decisiones, por lo que también es común ver en la literatura este tipo de modelos. Por ejemplo en Saeedi Mehrabad et al. (2017) se presenta un problema de ubicación jerárquica de fábricas y almacenes en una cadena de suministros de cuatro niveles con múltiples objetivos. Las principales características del modelo incluyen la determinación del número y la ubicación de las fabricas requeridas, el flujo de la materia prima de los proveedores a las fábricas, la determinación del número y la ubicación de los centros de distribución, así como el flujo de materiales de las fábricas a los centros de distribución, y finalmente la asignación de clientes a centros de distribución. Otro problema con decisiones jérarquicas puede verse en Ma et al. (2019) donde se presenta un problema de localización y asignación de almacenes para suministros en casos de desastres. Se sabe que en esos casos, es necesario la interacción de diferentes niveles de asignación de suministros, esto para aprovechar la integración de los recursos. El objetivo es minimizar el número total y el

costo de ubicación de almacenes y la distancia total para entregar suministros en todos los niveles.

Por otro lado, en Jiao et al. (2018) se estudia un modelo de optimización multi-objetivo para la localización de almacenes. Este modelo es motivado por los requerimentos de eficiencia de trabajo y seguridad del almacén automatizado. Dentro del modelo, se toman como funciones objetivo el tiempo de operaciones salientes-entrantes, el centro de gravedad de la plataforma general y el grado de acumulación relativa de productos. Continuando con los problemas multi-objetivo, en Ghodratnama et al. (2019) puede verse un desarrollo de planes de producción óptimo en municipios industriales, en donde se propone modelar la situación como modelos localización y asignación de centros. Para este caso los nodos concentradores se consideran como municipios industriales donde se encuentran fábricas y un almacén de distribución central. Se toman en cuenta dos objetivos: el primero minimizar los costos totales que incluyen el costo de la implementación de fábricas y almacenes, transporte, etc. y el segundo es minimizar el tiempo total transcurrido de productos en plantas de fabricación y almacenes.

En Izdebski et al. (2018) se modela un problema multi-criterio de localización de almacenes en una red logística. Los criterios de optimización involucrados son los correspondientes al costo de transporte, el costo asociado con los almacenes, esto es, los impuestos locales, gastos de arranque del almacén, el costo de abrirlo, los costos de la mano de obra, los costos de compra del terreno adicional para la expansión y los costos de transición de la materia prima a través de los almacenes. Para su solución se utiliza un algoritmo genético. Otro problema que considera los mismo criterios es el de Emeç and Akkaya (2018) pero a diferencia del anterior, el propósito es desarrollar un enfoque estocástico de toma de decisiones para solucionar el problema de ubicación de almacenes con condiciones de incertidumbre.

Además de problemas multi-criterio, también podemos encontrar problemas de decisiones tomadas por etapas, como puede observarse en Özceylan et al. (2017), donde se presenta un caso de estudio de de localización de almacenes de farmacias en la ciudad de Gaziantep. Primero se consideran las localizaciones actuales y posibles de los almacenes para proveer una distribución óptima para la distribución a hospitales y farmacias. Luego, se usa un modelo de conjunto de cobertura para determinar la capacidad de cobertura de los almacenes de farmacia actuales y potenciales y minimizar el número de almacenes que se abrirán. Y finalmente se usan modelos de la p-mediana y p-centros para abrir los almacenes potenciales y asignar farmacias y hospitales a los almacenes abiertos tal que la distancia total y la distancia más larga de la demanda al almacén se minimizan.

Después de revisar los problemas de localización de almacenes es lógico pensar en el siguiente paso de la cadena de suministros. En esta etapa, los almacenes que se localizan deben realizar procesos para minimizar sus costos de inventarios, traslados, pedidos, etc. En particular en el área de inventarios existen diversas políticas que se acoplan a las necesidades y enfoques de las empresas. En la siguiente sección se presenta un pequeño resumen sobre la teoría de inventarios.

2.3 Problemas de inventarios

Uno de los problemas más estudiados dentro de la teoría de cadenas de suministros es el problema de inventarios. A la cantidad de producto que la empresa tiene disponible para suministrar a sus clientes se le conoce como inventario. Esta cantidad de producto es muy importante ya que cuando se tiene una gran cantidad de inventario sin tener suficiente demanda de los clientes para ser vendido o entregado acarrea grandes costos para la empresa. Lo anterior se debe a que se tienen que almacenar los productos. Por otro lado, la falta de productos ante la gran demanda de los clientes también incurre en pérdidas para las empresas. Además disminuye la atracción de los clientes.

Existen diferentes enfoques de este tipo de problemas dependiendo de las características de los productos que se manejen en las empresas. Algunas veces es posible considerar que los clientes pueden esperar un poco más por la disponibilidad de un producto no esencial. Con esto se puede disminuir los costos de producción al tener pedidos más pequeños por hacer. Mientras que por otro lado, en los casos de salud, esta espera no es admisible. Tomando en cuenta estas características se pueden considerar diferentes políticas de inventarios. Los distintos problemas que se pueden encontrar de aplicaciones en la literatura parten de las premisas planteadas en la teoría. En la Tabla 2.1 puede verse un resumen de las suposiciones de los modelos determinísticos de la cantidad del pedido a realizar (EOQ por sus siglas en inglés).

Tabla 2.1: Modelos de inventarios determinísticos

	EOQ básico	**EOQ con faltantes planeados**	**EOQ con descuentos por cantidad**	**Modelos con revisión periódica**	**Modelos con múltiples escalones**
Demanda	conocida	Conocida	Conocida	Conocida	Varía dependiendo del escalón en el que se encuentre
Cantidad ordenada	Llega de una sola vez cuando se desea	Llega de una sola vez cuando se desea	Llega de una sola vez cuando se desea	Se revisa constantemente	varía dependiendo del escalón en el que se encuentre
Faltantes	No se permiten	Cuando ocurre un faltante, los clientes afectados esperan que el producto esté nuevamente disponible	No se permiten	No se permiten	Permitidos
Costo por unidad pedida	Homogéneo	Homogéneo	El costo unitario de un artÃculo depende de la cantidad de unidades que integren el lote	Homogéneo	Varía dependiendo del escalón en el que se encuentre

Un aspecto a considerar dentro de los problemas de inventarios es que la demanda de los clientes no siempre es la misma en diferentes intervalos de tiempo, o que los periodos de reorden no siempre son del mismo tamaño, por lo que los problemas presentados en la tabla anterior, también pueden estudiarse considerando algunos factores estocásticos.

Algunos trabajos que involucran decisiones de inventarios pueden verse en Battini et al. (2014) donde se presentan modelos tradicionales de inventarios que involucran diferentes decisiones con el objetivo de optimizar los tamaños de lote de materiales maximizando el costo total anual de la cadena. En ese trabajo se explora la integración de factores que afectan el impacto ambiental en conjunto con el modelo tradicional EOQ y además se propone un modelo EOQ sustentable. Todos los factores considerados están ligados al tamaño de los lotes de materiales y analizados desde el principio de órdenes de compra de materia prima hasta el final de su vida dentro de las plantas.

Por otro lado en Matheus and Gelders (2000), se considera un problema de inventario sujeto a patrones de demanda probabilística de diferentes tamaños, y se propone una forma exacta y una por aproximación de calcular el punto de reorden para la política de inventarios (R, Q). El algoritmo exacto involucra fórmulas para la distribución discreta de la demanda total durante el tiempo de reabastecimiento. Mientras que el método de aproximación es basado en el uso de distribuciones continuas. Otro problema de tipo (R,Q) se propone en Guan and Zhao (2011). Para ese caso, se considera un sistema con múltiples clientes operado en un horizonte infinito, en el cual cada uno de los clientes tiene demanda estocástica que sigue un proceso de Poisson adoptando una política de revisión continua (R,Q) para satisfacer la demanda de los clientes. El sistema involucra decisiones de precios y manejo de inventarios, el objetivo es maximizar el beneficio, el cual es igual a las ventas menos el costo de compra y de inventario.

Modelos que involucran múltiples escalones de la cadena de suministros pueden verse en la revisión de literatura de Gümüs and Güneri (2007), donde se remarcan algunos trabajos como los de Sherbrooke (1968) donde se considera una política de ordenamiento de un modelo de dos escalones en donde se consideran almacenes y clientes. En ese trabajo se asume que puede haber faltantes en los pedidios. Además, en ese mismo trabajo se contruye la métrica que identifica los niveles de existencia que minimizan el número esperado de pedidos pendientes en el escalón inferior sujetos a restricciones de presupuestos. Este es considerado uno de los primeros trabajos modelados para múltiples escalones.

A partir de entonces, una gran cantidad de modelos, que generalmente buscaban identificar tamaños de lote óptimos considerando inventarios de seguridad utilizando múltiples escalones de la cadena de suministros pueden hallarse en la literatura.

Como puede verse, las decisiones de inventarios son de suma importancia dentro de la cadena de suministros, por lo que para este problema fue considerado en conjunto con el problema de localización. En la siguiente sección, se presenta una revisión de literatura sobre problemas que han considerados políticas de inventarios durante la decisión de localización de instalaciones.

2.4 Problemas de localización con decisiones de inventarios

Los problemas de localización y asignación pueden verse en combinación con otras decisiones de la cadena de suministros tales como ruteo, clusterización, producción e inventarios. En esta sección se presenta un resumen de los problemas que combinan la decisión de localización con la de inventarios.

Por ejemplo, en Yao et al. (2010) se presenta un problema motivado por una situación real que ocurre en una compañía de química aplicada, en la cual se producen diferentes artículos en distintas plantas. Los almacenes pueden ser surtidos por distintas plantas al mismo tiempo debido a la capacidad de ellas. También se considera que las demandas de los clientes son estocásticas y en los almacenes se cuenta con pedido de seguridad. El objetivo del problema es determinar el número y la localización de los almacenes, además de la asignación de la demanda de los clientes y nivel de inventario de cada almacén. El problema es formulado como un modelo de programación entero mixto no lineal y utiliza técnicas de aproximación y transformación para desarrollar un heurístico iterativo para obtener soluciones de buena calidad.

Otro problema de localización con inventarios se presenta en Mousavi et al. (2015), donde se propone un modelo no lineal entero mixto. Dicho modelo pretende encontrar el número óptimo de productos a comprar por los clientes a distribuidores en diferentes periodos. Además, se determina la localización de los distribuidores buscando minimizar el costo total de inventarios. Para resolver este modelo se presenta un algoritmo modificado basado en la metaheurística de la mosca de la fruta. Por otro lado, en Cabrera et al. (2013) se presenta un problema de localización con inventarios con restricciones de capacidades estocásticas basadas en una política de control periódico de inventario. El problema es resuelto por medio de búsqueda tabú y un algoritmo de optimización de enjambre de partículas debido a la complejidad del problema en cuestión. En Daskin et al. (2002) se incorporan las decisiones de localización e inventarios de centros de distribución. El problema es formulado con un modelo de programación entera no lineal y se utiliza un

algoritmo de relajación Lagrangiana para resolver el modelo explotando la estructura del modelo.

Una aplicación del problema de localización que considera la restricción de abrir p instalaciones considerando inventarios puede verse en Xu (1999), dicho problema es motivado por la forma de distribuir productos perecederos. En este caso se debe localizar un número prestablecido de bancos de sangre. En el modelo se minimiza la suma de los costos fijos, transporte y efecto de inventario de riesgo. El modelo es resuelto usando un algoritmo de sustitución con un pequeño conjunto de datos de prueba.

Los problemas antes referidos han sido considerados bajo el supuesto de que hay un solo tomador de decisiones que se encarga de decidir la localización y la política de inventarios. Sin embargo, también existen trabajos en donde se consideran a dos tomadores de decisiones con cierta jerarquía predefinida entre ellos. Dichos problemas han sido modelados como problemas de programación binivel, por ejemplo en Wang et al. (2007) se considera el problema de una cadena de suministros con un proveedor, un centro de atención negocio-cliente y múltiples centros de distribución. En conjunto analizan la localización y políticas de inventario cuando hay devolución de productos. Una nueva política de localización e inventarios se estudia y es modelada como un problema binivel en donde en el nivel superior se determina la localización de los sitios terceros de control; y en el nivel inferior se toma la decisión de una política de reposición de inventarios coordinada QS_R que se debe considerar en la localización de los sitios terceros de control.

El problema presentado en este trabajo de tesis puede verse como una variante del problema estudiado en Cabrera et al. (2013), donde se considera un problema de dos escalones de la cadena de suministros. Los autores presentan un problema donde la compañía necesita localizar almacenes y asignar los clientes a ellos considerando costos tales como los de instalación, transporte, inventario y de colocar órdenes. Todos estos costos se consideran dentro de la misma función objetivo. Para esta tesis la compañía del mayor nivel jerárquico solo toma la decisión de localizar los almacenes y asignar los clientes a ellos. Por otro lado, cada almacen raliza su propia política de inventarios considerando su beneficio, esto es, minimizar el costo total de colocar una orden, considerando la demanda total de los clientes asignados a él. Una vez que la orden es calculada por el almacén, esta cantidad tiene un impacto para la compañía, añadiendo un costo extra a la decisión de localización.

Para nuestro problema en cuestión, se toman ambas decisiones dentro del mismo modelo, pero de una forma jerárquica, lo que nos lleva a tener un problema de programacón

binivel. A continuación se presenta la teoría sobre los problemas binivel para tener una mejor idea de la formulación del problema.

2.5 Programación Binivel

Un Problema de Programación Binivel (PPB) es aquel en donde una de sus restricciones está representada por otro problema de optimización. Es decir, se tienen dos problemas interrelacionados entre sí, uno en el nivel superior (asociado a un líder) y uno en el nivel inferior (asociado a un seguidor). Esquemáticamente puede verse como que en cada nivel se debe tomar una decisión que afectará al otro creando una relación jerárquica entre líder y seguidor. Es importante notar que este tipo de problemas son diferentes a los bi-objetivo debido a la jerarquía preestablecida y a que un subconjunto de las variables de decisión están implícitamente determinadas por la solución óptima de otro problema.

El modelo matemático para formular un PPB fue introducido por primera vez en Bracken and McGill (1974), en donde abordaron un problema de asignación de recursos y armamento para optimizar la ofensiva y defensiva militar. La formulación general de un PPB propuesta en Bracken and McGill (1974) es la siguiente:

$$\begin{aligned} &\min_{x \in X, y} F(x, y) \\ s.a.\ &G(x, y) \leq 0 \\ &\min_{y} f(x, y) \\ s.a.\ &g(x, y) \leq 0 \end{aligned}$$

Donde $x \in \mathbb{R}^{n_1}$ y $y \in \mathbb{R}^{n_2}$. Las variables del problema se dividen en dos clases llamadas: variables del nivel superior ($x \in \mathbb{R}^{n_1}$) y variables del nivel inferior ($y \in \mathbb{R}^{n_2}$). Por otro lado, la función $F : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$ y $f : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$ son las funciones objetivo del nivel superior e inferior, respectivamente; mientras que las funciones $G : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_1}$ y $g : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{m_2}$ son las restricciones del nivel superior e inferior, respectivamente.

Este tipo de problemas son complejos de resolver. Para evidenciar esto, es suficiente con observar un PPB lineal, el cual es la versión más simple de este tipo de problemas. En

Jeroslow (1985) se mostró que un PPB lineal se clasifica como NP-Hard, tiempo después en Hansen et al. (1992) se demostró que es fuertemente NP-Hard.

Aunque los PPB son complejos de resolver, su estructura facilita la formulación de un gran número de problemas que involucran procesos de decisión jerárquica. Revisiones de literatura muy completas se pueden encontrar en Kalashnikov et al. (2015); Vicente and Calamai (1994); Wen and Hsu (1991); Colson et al. (2007). En dichos trabajos se muestran propiedades de los PPB, formas de resolverlos y algunas aplicaciones tales como los modelos de migración, planeación, asignación, protección de información, transporte, diseño de ingeniería, logística humanitaria, fijación de cuotas, entre otros.

A continuación se describen algunos problemas binivel de localización en conjunto con otras decisiones, en entre ellas las de inventario. Además, se mencionan los métodos propuestos para su resolución.

Un problema de la p-mediana puede encontrarse en Abareshi and Zaferanieh (2019), donde se presenta un modelo binivel para evaluar el problema de localización de instalaciones con capacidades. Para el problema del nivel superior se considera el problema clásico de la p-mediana pero con capacidades para minimizar el costo de localizar las instalaciones y satisfacer las demandas de los clientes. Mientras que el modelo del nivel inferior se presenta un modelo basado en registros para determinar la asignación más probable basada en la información disponible. El problema se resuelve utilizando la teoría dual lagragiana. Dicha teoría es utilizada para reducir el problema binivel a un problema entero mixto no lineal de un nivel cuya solución se obtiene al comparar dos problemas lineales mixtos.

En Calvete et al. (2011) se presenta un problema de producción y distribución. En ese trabajo se consideran dos tomadores de decisiones controlando los procesos de producción y la distribución. La compañia encargada de la distribución es el líder del proceso, y controla la asignación de los clientes a los centros de distribución y la ruta que los servirá. Por otro lado, la compañia encargada de la manufactura es el seguidor, quien decide cuáles plantas fabricarán los productos. El problema es modelado como un probema de programación binivel y se utiliza un algoritmo de colonia de hormigas para su solución. Un problema similar se presenta en Marinakis and Marinaki (2008), donde se aborda un problema generado en una empresa encargada de distribución de madera en Grecia. Se propone un nuevo modelo de localización y ruteo. Basado en el hecho que

las decisiones se toman a nivel estratégico y a nivel operativo, se formula el problema de tal manera que en el primer nivel se toman las decisiones del nivel estratégico, es decir, el gerente superior encuentra la ubicación óptima de las instalaciones; mientras que en el segundo nivel se toman las decisiones de operación, es decir, el gerente operativo encuentra la ruta óptima de los vehículos. Para este problema, se propone un algoritmo genético.

Además, en Kongsomsaksakul et al. (2005) y en Chen, Tadikamalla, Shang and Song (2020) se presentan trabajos de logística humanitaria. En Kongsomsaksakul et al. (2005) se estudia un problema de localización de refugios en casos de inundaciones. El problema se plantea como un juego de Stackelberg, que consiste en que una autoridad (el líder) determina la ubicación de los refugios para minimizar el tiempo total de evacuación; mientras que los evacuados (seguidores) eligen el destino y la ruta por la cual serán evacuados. Para la solución del problema se utiliza un algoritmo genético. Para Chen, Tadikamalla, Shang and Song (2020) el nivel superior presenta un problema de localización de centros de emergencia y la programación de la forma en que se van a repartir los suministros hacia los centros de emergencia. Por su parte en el nivel inferior, se toma la decisión de asignar los suministros de los centros de emergencia a los lugares afectados tomando en cuenta que se haga de forma equitativa.

En Ghasemi and Khalili-Damghani (2021) también se estudia un problema de logística humanitaria, pero el problema se modela de forma trinivel. El primer nivel determina el número y la localización de los centros de distribución. En el segundo nivel se decide la cantidad de flujo de productos básicos críticos de los proveedores a los centros de distribución y luego a las áreas afectadas. Mientras que en el tercer nivel se consideran los niveles óptimos de inventario en cada uno de los centros de distribución. Para su solucón también se utiliza un algoritmo genético.

En CUI and LI (2007) se puede ver un problema de localización e inventarios para una red de distribución de dos escalones de la cadena de suministros considerando una política multiperiodo. El objetivo es localizar centros de distribución para después programar las rutas y determinar las cantidades de pedidos de los clientes. Para resolver este problema se propone un algoritmo heurístico diseñado especialmente para este problema.

En particular, el problema considerado en esta tesis doctoral es modelado como un problema de programación binivel con múltiples seguidores independientes. Debido a la naturaleza jerárquica del proceso de toma de decisiones, la empresa (de aquí en adelante

será referida como el líder) será la encargada de localizar los almacenes y asignar los clientes a ellos. Después de eso, cada uno de los almacenes (de aquí en adelante serán referidos como los seguidores) deben tomar su propia decisión de inventarios. La cantidad de pedidos que se realice en cada uno de los almacenes impacta en un costo extra para la empresa al momento de localizar los almacenes. Es conveniente mencionar que este problema puede verse como un problema binivel con múltiples seguidores independientes ya que cada almacén tiene asignados sus propios clientes los cuales no interfieren en la cantidad de pedido que se realizará en los otros almacenes.

Capítulo 3

Descripción del problema binivel de localización que considera políticas de inventarios.

En este capítulo se va a explicar a detalle la problemática bajo estudio. Luego, se presenta el modelo matemático propuesto para modelar dicha situación. Por último, se describen algunas características especiales del modelo de programación binivel resultante. Dichas características serán explotadas por el algoritmo descrito en el Capítulo 4.

3.1 Planteamiento del problema

Considere un conjunto de sitios en los cuales pueden ser abiertos algunos almacenes para dar servicio a un conjunto de clientes. En el nivel superior (ver Figura 3.1), el líder intenta minimizar los costos de abrir los almacenes y asignar los clientes a ellos. Para lograr esto, considera como función objetivo el costo de abrir los almacenes y el costo de asignar los clientes a ellos. Al tomar esa decisión se debe garantizar la apertura exacta de los almacenes que la empresa requiere. Es claro ver que no puede haber clientes sin ser asignados a almacenes, ni almacenes abiertos sin clientes ya que ésto se traduciría para la empresa en unos costos innecesarios.

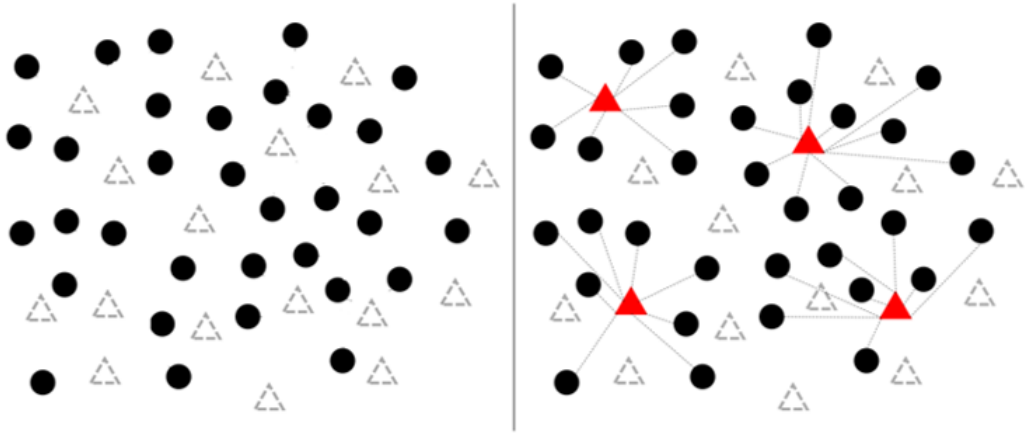


Figura 3.1: Nivel superior de modelo binivel. Ejemplo con 33 clientes (círculos e color negro) y 15 sitios potenciales para ubicar los almacenes (triángulos) de los cuales se requiere abrir 4 (triángulos en color rojo).

Una vez que el líder ha decidido la localización de los almacenes y ha asignado los clientes a ellos, cada uno de los almacenes (seguidores) se encarga de realizar su propia política de inventario (ver Figura 3.2). Para definir esta política de inventario, los almacenes consideran que puede haber pedidos atrasados, más no deben exceder cierto límite. Además, cada uno de los almacenes cuenta con una capacidad para atender los pedidos. Obviamente, dicha capacidad no debe sobrepasarse. Los costos por colocar pedido, pedidos atrasados e inventarios son distintos en cada uno de los almacenes. También se considera que las demandas de los clientes son fijas, esto quiere decir que los clientes tienen la misma demanda en cualquier intervalo de tiempo.

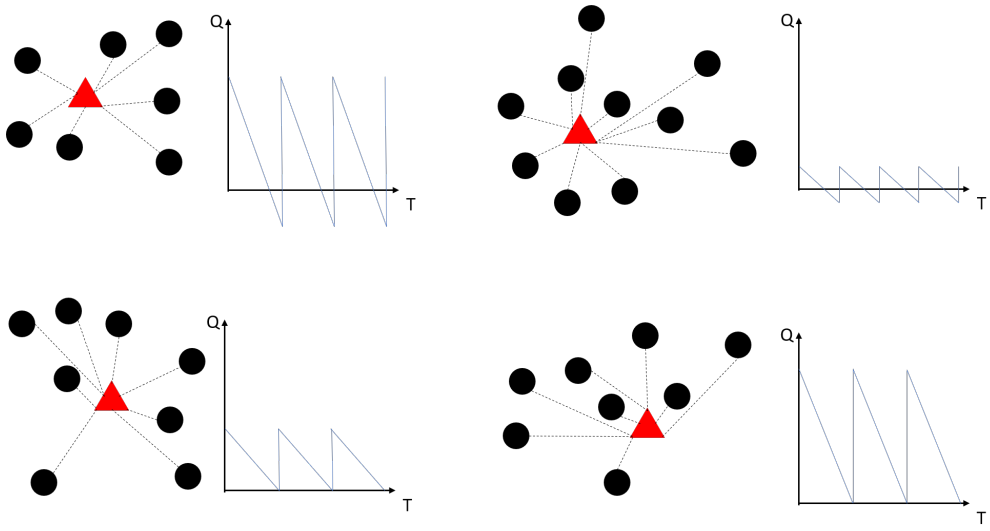


Figura 3.2: Nivel inferior de modelo binivel. Ejemplo con 4 almacenes, donde cada uno decide de forma independiente la cantidad de pedido a realizar.

El hecho de que cada uno de los almacenes considere sus propios costos permite analizar cada problema del nivel inferior de manera independiente. Además, los costos no son directamente proporcionales a la distancia que tienen los almacenes con los clientes, sino mas bien dependen de la forma en que se lleva a cabo el proceso de producción. Esta cantidad de pedido es tomada en cuenta por el líder al momento de realizar la localización ya que incurre en un costo extra para él, el cual puede ser reflejado como un costo de traslado de los pedidos a los almacenes.

3.1.1 Modelación Matemática

En esta sección, se presenta la formulación matemática del problema aquí estudiado. Como se mencionó anteriormente, este problema es modelado como un programa binivel con múltiples seguidores independientes. A continuación se describen todos los componentes involucrados en el modelo; es decir, los conjuntos, parámetros y las variables de decisión.

CONJUNTOS

$I = \{1, 2, 3, \ldots, n\}$ Conjunto de índices de los almacenes.

$J = \{1, 2, 3, \ldots, m\}$ Conjunto de índices de los clientes.

PARÁMETROS

p : Número de almacenes que deben ser abiertos.

F_i : Costo de apertura de un almacén.

c_{ij} : Costo de asociar clientes a los almacenes.

d_j : Demanda de cada cliente.

β_i : Número máximo de faltantes en cada almacén.

I_i^{cap} : Capacidad de cada almacén justo después de que llegue un pedido.

h_i : Costo de tener una unidad en inventario.

π_i : Costo por órdenes atrasadas por unidad de producto.

$\hat{\pi}_i$: Costo por órdenes atrasadas por unidad de tiempo.

VARIABLES DE DECISIÓN

Las variables de decisión del líder son:

$$x_i = \begin{cases} 1 & \text{si el almacén } i \text{ es abierto} \\ 0 & \text{en otro caso} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si el cliente } j \text{ está asociado con el almacén } i \\ 0 & \text{en otro caso} \end{cases}$$

para cada seguidor, las variables de decisión correspondientes son:

Q_i = Cantidad de pedido realizada en cada almacén i por ciclo.

B_i = Nivel de órdenes atrasadas justo después de recibir una orden de Q_i unidades.

El modelo de programación bi-nivel resultante para el problema en cuestión se presenta a continuación:

$$\min_{x,y} \sum_{i \in I} \left[F_i x_i + \sum_{j \in J} c_{ij} y_{ij} + o_i Q_i \right] \tag{3.1}$$

sujeto a:

$$\sum_{i \in I} x_i = p, \tag{3.2}$$

$$\sum_{i \in I} y_{ij} = 1, \quad j \in J \tag{3.3}$$

$$y_{ij} \leq x_i, \quad i \in I, j \in J \tag{3.4}$$

$$x_i, y_{ij} \in \{0, 1\} \tag{3.5}$$

donde para cada almacén i abierto se resuelve el siguiente problema:

$$\min_{Q,B} \sum_{j \in J} \left[\frac{A_i}{Q_i} y_{ij} d_j + \frac{x_i y_{ij} \pi_i B_i d_j}{Q_i} + \frac{x_i h_i (Q_i - B_i)^2}{2Q_i} + \frac{x_i \hat{\pi}_i (B_i)^2}{2Q_i} \right] \tag{3.6}$$

$$Q_i - B_i \leq I_i^{cap} x_i, \qquad i \in I \tag{3.7}$$

$$B_i x_i \leq \beta_i \qquad i \in I \tag{3.8}$$

$$Q_i > 0, \qquad B_i \geq 0 \tag{3.9}$$

En el problema del líder, la compañía necesita localizar (abrir) almacenes para después asociar clientes a cada uno de los almacenes que son abiertos considerando minimizar el costo total como se muestra en la Ec. (3.1). Para garantizar la apertura de exactamente p almacenes se tiene la restricción Ec. (3.2). Algo que es tomado en cuenta cuando la compañía localiza los almacenes es que los clientes estén asociados a un solo almacén. Esta restricción puede verse en la Ec. (3.3). Otra de las consideraciones a tomar en cuenta es que los clientes estén asociados solo a almacenes abiertos como se muestra en la Ec. (3.4). En la Ec. (3.5) se define la naturaleza binaria de las variables del líder.

Una vez que los clientes son asignados a los almacenes. Cada almacén abierto tiene el objetivo de minimizar su costo de pedido, esto se exige con la Ec (3.6). Para garantizar que la capacidad de los almacenes no sea excedida cuando una nueva orden llega se incluye la restricción dada en la Ec. (3.7). Como el hecho de incurrir en faltantes trae un costo extra, cada almacén tiene un máximo número de órdenes atrasadas permitidas, tal y como se considera en Ec. (3.8). Por último, se presenta la naturaleza de las variables del seguidor en la Ec. (3.9).

Es importante hacer notar que cuando $x_i = 0$, es decir, cuando un almacén no fue abierto, entonces se tiene que $Q_i = 0$, lo cual indetermina la función objetivo en el nivel inferior. En este caso, el nivel inferior del modelo puede reescribirse de la siguiente forma:

$$\min_{Q,B} \sum_{i\in\bar{I}} \sum_{j\in J} \left[\frac{A_i}{Q_i} y_{ij} d_j + \frac{x_i y_{ij} \pi_i B_i d_j}{Q_i} + \frac{x_i h_i (Q_i - B_i)^2}{2Q_i} + \frac{x_i \hat{\pi}_i (B_i)^2}{2Q_i} \right] \tag{3.10}$$

$$Q_i - B_i \leq I_i^{cap} x_i, \qquad i \in \bar{I} \tag{3.11}$$

$$B_i x_i \leq \beta_i \qquad i \in \bar{I} \tag{3.12}$$

$$Q_i > 0, \qquad B_i \geq 0 \tag{3.13}$$

donde $\bar{I} = \{i \in I : x_i = 1\}$, es decir, el conjunto de índices de los almacenes que se han abierto en el nivel superior.

Debido a la forma no lineal que presenta el nivel inferior se puede emplear el método de los multiplicadores de Lagrange (ver apéndice A) para obtener una solución óptima de ese problema. Lo anterior es crucial a la hora de diseñar un algoritmo de solución para el problema bi-nivel.

Capítulo 4

Algoritmos de solución propuestos

Como se mencionó en secciones anteriores, el problema propuesto en este trabajo de tesis no tiene antecedentes de modelado ni de métodos de solución. Nosotros proponemos un algoritmo de lluvia de ideas para resolverlo. La motivación para seleccionar este algoritmo es basada en experiencias previas exitosas para la resolución de problemas binivel. Para tener un punto de comparación de las soluciones del algoritmo de lluvia de ideas propuesto, se diseñó y utilizó también un algoritmo genético. El algoritmo genético fue seleccionado debido a la similitud de ciertas partes con el algoritmo de lluvia de ideas. Ambos algoritmos metaheurísticos cuentan con buenos antecedentes para resolver problemas binivel (ver Abo-Elnaga et al., 2021; Luo et al., 2021; Nucamendi-Guillén et al., 2018; Yuan et al., 2021; Barman and Das, 2021). Además de los dos algoritmos mencionados anteriormente, para contar con algunas soluciones óptimas (cuando fue posible), se hizo uso de un algoritmo exhaustivo. Es bien sabido que este tipo de algoritmos son poco eficientes en cuanto a tiempos computacionales y la mayoría de las veces, debido al tamaño de los problemas, no logran resolverlos porque las computadoras llegan a agotar su memoria. Aún así, viéndolo desde un punto comparativo, puede ser una buena opción para problemas de tamaño pequeños que se mencionarán más delante. En la primera parte de este capítulo se presenta una pequeña introducción de los algoritmos evolutivos, después se introduce el algoritmo de lluvia de ideas propuesto para este problema, luego, se describe el algoritmo genético, y por último, se explica el algoritmo exhaustivo.

4.1 Algoritmos evolutivos

Los algoritmos de optimización basados en poblaciones han sido aceptados y exitosamente aplicados para resolver una gran variedad de problemas de optimización. A diferencia de los algoritmos basados en una única solución, los algoritmos de optimización basados en poblaciones consisten en un conjunto de soluciones (población) que resuelven el problema a través de información compartida para cooperar y competir entre ellas. Ac-

tualmente, existe una gran variedad de este tipo de algoritmos. Los primeros algoritmos propuestos fueron aquellos que se basan en la evolución biológica, tal es el caso de los algoritmos genéticos (ver Mirjalili, 2019; Katoch et al., 2021; Kim et al., 2021; Hamdia et al., 2021) y los evolutivos (ver Aggarwal et al., 2021; Del Prete et al., 2020; Godini et al., 2021; Anton, 2020) como se muestra en la Figura 4.1.

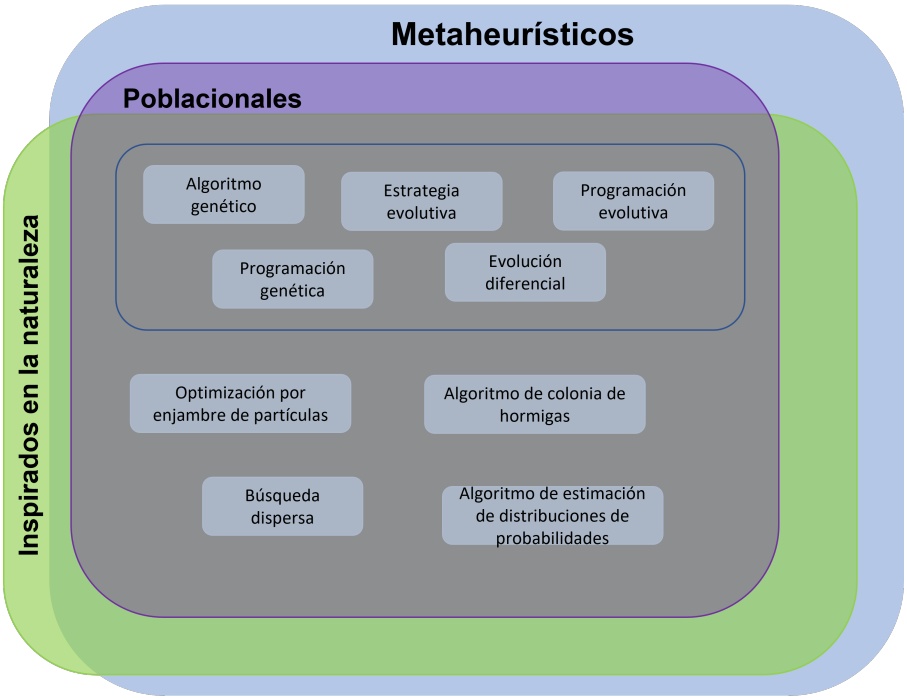


Figura 4.1: Algunos algoritmos metaheurísticos poblacionales

Recientemente, se han desarrollado más algoritmos basados en poblaciones que son usualmente llamados "inspirados en la evolución natural". Muchos de estos algoritmos son clasificados como algoritmos de enjambres inteligentes. En este tipo de algoritmos, cada individuo en la población representa un objeto; por ejemplo, una hormiga, pájaro, pez, etc. En un algoritmo de enjambre inteligente, es el comportamiento colectivo de todos los individuos lo que hace que el algoritmo sea eficaz en la optimización de problemas. Todos los individuos cooperan y avanzan colectivamente hacia las mejores áreas en el espacio de búsqueda de soluciones. Existe una gran variedad de algoritmos de enjambre inteligentes, de entre los cuales sobresalen: el algoritmo de enjambre de partículas (ver Piotrowski et al., 2020; Wang, Zhang and Zhou, 2021; Zhang et al., 2020; Wang et al., 2020), el algoritmo de colonia de hormigas (ver Paniri et al., 2020; Zhao et al., 2021; Yang et al., 2020), el algoritmo de optimización bacteriano (ver Farshi and Orujpour, 2021; Chen, Wang, Di and Ping, 2020; Muni et al., 2021; Mahmoud et al., 2020), el algoritmo

de optimización de luciérnagas (ver Devaraj et al., 2020; Sennan et al., 2020; Dai et al., 2020; Aydilek et al., 2021), entre otros.

Perteneciente a los algoritmos de enjambre de partículas, recientemente, se ha propuesto el algoritmo de optimización de lluvias de ideas. En la siguiente sección nos enfocaremos en presentar sus antecedentes y su adaptación al problema estudiado.

4.2 Algoritmo de lluvia de ideas para resolver el problema binivel bajo estudio

Cuando nos enfrentamos a un problema difícil, por ejemplo, aquel que una sola persona no puede resolver, es común recurrir a la técnica de lluvia de ideas. Esta técnica consiste en reunir a un grupo de personas con diferentes puntos de vista sobre el problema. Luego, dichas personas generan una variedad de ideas que generalmente, en conjunto, resuelven el problema con una alta probabilidad.

En un proceso de lluvia de ideas, normalmente hay un moderador, un grupo de personas que propician la lluvia de ideas y varios expertos en el problema a resolver. El papel del moderador es promover el proceso de generación de ideas (lluvia de ideas) basado en las cuatro reglas de Osborn Osborn (1957), las cuales se resumen en la Tabla 4.1. El moderador no debe participar en la generación de ideas en sí, por lo que se recomienda que sea alguien que desconozca del problema a solucionar para así evitar el menor sesgo posible.

Regla 1: Evitar el juicio	*Regla 2: Cualquier idea es válida*
Regla 3: Cruza de ideas	*Regla 4: Ir por la cantidad*

Tabla 4.1: Reglas propuestas por Osborn para el proceso de lluvia de ideas.

En la Tabla 4.1, la regla 1 denota que no hay ideas malas. No es prudente juzgar si una idea propuesta es buena o mala. Cualquier juicio o críticas deben guardarse hasta el final del proceso de lluvia de ideas. La regla 2 dice que cualquier cosa que ocurra durante el proceso de lluvia de ideas es una idea que vale la pena compartir y grabar. Ninguna idea debe ser ignorada. La regla 3 dice que muchas ideas pueden y deben basarse en ideas ya generadas. Cualquier idea generada puede y debe servir como pista para generar más ideas. La regla 4 menciona que es necesario generar tantas ideas como sea posible. Lo primero que se debe buscar es tener una buena cantidad de ideas. La calidad viene de la

cantidad de forma natural. Si no se tiene una buena cantidad de ideas, es difícil identificar ideas de buena calidad. El propósito de generar ideas de acuerdo a las reglas mencionadas en la Tabla 4.1 es generar ideas tan diversas como sea posible, de tal forma que las personas involucradas en la lluvia de ideas puedan abrir su mente lo máximo posible. El diverger a ideas previa generadas, ayuda a no quedarse atrapados en las mismas ideas. Por último, los expertos del problema sirven para un propósito diferente. Son los encargados de tomar varias buenas ideas de las que han sido generadas, y con esto motivar a que el grupo de lluvia de ideas considere como base a las mejores ideas para generar futuras ideas.

El proceso de lluvia de ideas anteriormente descrito, ha sido exitosamente aplicado para resolver problemas muy complejos, debido a esto, en Shi (2011) se consideró que un algoritmo cuyo diseño se basara en este proceso debía ser de igual o de mejor calidad que los inspirados en el comportamiento natural de los animales. El algoritmo propuesto en Shi (2011) considera que cada solución al problema a resolver representa una idea, entonces al igual que en el proceso de lluvia de ideas, se debe contar con una cantidad de ideas inicial, después, las ideas son separadas en grupos de acuerdo a cierta característica particular. Para la generación de nuevas ideas se utilizan operadores de combinación y perturbación. Por último, se actualiza la población incluyendo las mejores ideas de la población previa y las mejores ideas nuevas.

A continuación se presenta el algoritmo de lluvia de ideas propuesto en Shi (2011).

1. Generar aleatoriamente n ideas (soluciones);
2. Agrupar las n ideas en m grupos;
3. Evaluar las n ideas;
4. Ordenar las ideas en cada grupo y colocar la mejor idea como el centro en cada grupo;
5. Aleatoriamente generar un valor entre 0 y 1;
   a) Si el valor es menor que una predeterminada probabilidad p_1,
      1) Aleatorimente seleccionar un centro de un grupo;
      2) Aleatoriamente generar un individuo para reemplazar el centro del grupo seleccionado.
6. Generar nuevas ideas;
   a) Aleatoriamente generar un valor entre 0 y 1;

b) Si el valor es menor que una probabilidad p_2,

1) Aleatoriamente seleccionar un grupo con probabilidad p_3,
2) Generar un valor entre 0 y 1;
3) Si el valor es menor que cierta probabilidad p_4,
   a' Seleccionar el centro de un grupo y cambiar aleatoriamente parte de la idea para generar una nueva idea.
4) De lo contrario, seleccionar aleatoriamente una idea de este grupo y cambiar aleatoriamente una parte de la idea para generar una nueva.

c) De lo contrario, seleccionar aleatoriamente dos grupos para generar una nueva idea.

1) Generar un valor aleatorio;
2) Si es menor a cierta probabilidad p_5, se combinan los dos centros de los dos grupos seleccionados para generar una nueva idea.
3) De lo contrario, dos ideas de los grupos seleccionados se combinan y de igual forma generar una nueva idea.

d) La nueva idea generada es comparada con las existentes pertenecientes al mismo grupo, las mejores son actualizadas como la nueva población de ideas.

7. Si n nuevas ideas fueron generadas, ir al paso 8; de otra forma regresar al paso 6;

8. El algoritmo termina si un número predefinido de iteraciones ha sido alcanzado. De lo contrario, ir al paso 2.

En el algoritmo anterior, el paso 1 es el mismo paso inicial utilizado en cualquier algoritmo poblacional, es decir, generar la población inicial. Los pasos 2., 3. y 4. sirven para seleccionar las mejores ideas. El paso 5. ayuda a explorar una mayor área del espacio de soluciones. En el paso 6. se lleva a cabo la generación de nuevas ideas. El paso 6.*b*) ayuda a generar nuevas ideas inspiradas en una sola idea ya existente, mientras que el paso 6.*c*) genera una nueva idea a partir de dos ideas pertenecientes a dos grupos diferentes. Los centros de los grupos tienen mayor probabilidad de ser usados en la generación de una nueva idea que las demás ideas pertenecientes al mismo grupo. El hecho de separar a las ideas en grupos, tiene el propósito de actuar como los diferentes expertos del problema; el centro de cada grupo sirve como la mejor idea seleccionada por el experto del problema. El paso 6.*d*) tiene el propósito de mantener a las mejores ideas generadas en las siguiente poblaciones. El tamaño de población n equivale al número de ideas generadas en cada ronda de generación de ideas en el proceso de lluvia de ideas.

En la figura 4.2, se muestra el diagrama general del algoritmo de lluvia de ideas utilizado para la solución del problema bi-nivel.

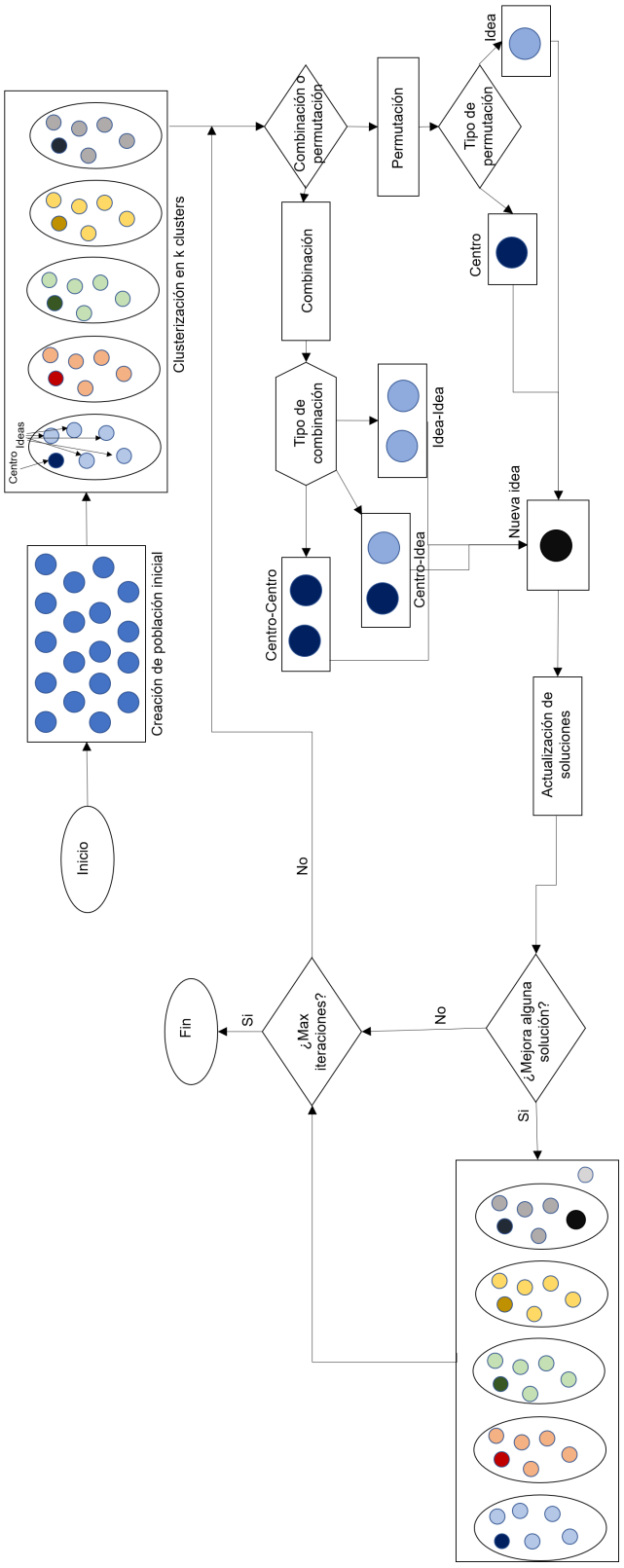


Figura 4.2: Diagrama del algoritmo de lluvia de ideas

Hay que tener claro que para obtener una solución del problema bi-nivel, hay que considerar que cada vez que se construye o actualiza una idea hay que resolver el problema del nivel inferior. Como se mencionó anteriormente, el problema del nivel inferior es no lineal, por lo que es dificíl resolver por medio de un optimizador comercial. Debido a esto, se hace uso del método de los multiplicadores de Lagrange para resolverlo. Este último método se resuelve en un tiempo computacional relativamente bajo, por lo que no causa problemas significativos al momento de la implementación del algoritmo (ver Apéndice A).

A continuación se revisará a detalle cada uno de los componentes involucrados dentro del algoritmo de lluvia de ideas para resolver el problema binivel.

4.2.1 Creación de población inicial

Como se observa en la primera parte del diagrama presentado en la Figura 5.1, el proceso inicia con la creación de una población inicial de ideas (soluciones). Para lo cual debe considerarse que se deben tener una parte de ideas variadas y una de calidad (en cuanto a función objetivo). Para garantizar generar ideas de calidad, se crea un pre-proceso en donde se generan ideas de forma voraz con respecto a los costos. Es evidente que si se consideran los mismos costos para generar varias ideas de esta manera, siempre se obtendrá la misma solución. Entonces, se generan costos ficticios de manera aleatoria respetando de cierto modo a los valores de los costos iniciales para evitar tener soluciones puramente aleatorias. Es decir, se considera un intervalo de variación en cuanto a los costos de apertura de los almacenes y asignación de los clientes. Por otro lado, para agregar diversidad en la población inicial se generan ideas de manera aleatoria.

Como cada una de las ideas representa una solución del problema binivel, entonces, éstas deben representar los sitios potenciales elegidos para abrir los almacenes, dichas soluciones se codifican en forma de vectores binarios (ver Figura 4.3) de longitud igual al número de sitios potenciales para abrir los almacenes, donde la entrada será igual a 1 si el almacén es abierto, o 0 si no (ver la parte del modelo del problema).

Figura 4.3: Representación de una idea con 7 sitios potenciales y 3 almacenes abiertos

Después de que se genera la población inicial de n ideas, se procede a realizar la agrupación. Para esto, se utiliza la función Hash que corresponde a cada idea, la cual se calcula de la siguiente forma:

$$\text{Hash(x)} = \sum_{i \in I} 2^i x_i \tag{4.1}$$

Donde i representa la posición del vector y x_i corresponde a la variable binaria del nivel superior.

Respecto a este valor, se etiquetan a las n ideas con respecto a la forma en que está distribuida la apertura de los almacenes. Es decir, las primeras ideas ordenadas tienen los primeros almacenes abiertos, mientras que las últimas ideas ordenadas tienden a tener los últimos almacenes abiertos. Esto se debe al valor Hash correspondiente. Una vez que las ideas han sido etiquetadas, se utiliza el método de la $k - medias$ Lloyd (1957) para agruparlas en k grupos con respecto a dicho valor.

Un ejemplo se puede ver en la Figura 5.1, en donde si se desean realizar dos grupos con las ideas n_1, n_2, n_3, y n_4 con el método de las $k-medias$. Las ideas n_1 y n_3 quedarían en el mismo grupo, mientras que las ideas n_2 y n_4 en otro. Además, como puede notarse, n_1 y n_3 tienden a tener los primeros almacenes abiertos, mientras que n_2 y n_4 tienen abiertos a los últimos almacenes.

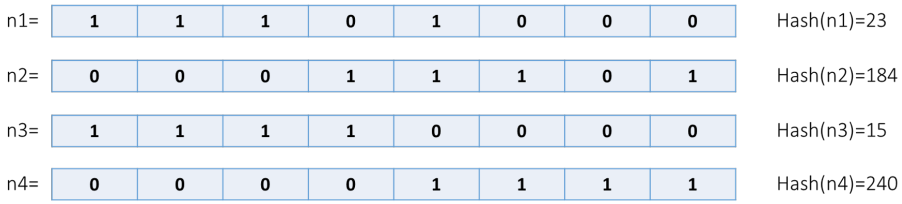


Figura 4.4: Valor de la función Hash para cuatro soluciones aleatorias

Cuando una idea n ha sido asignada a un grupo, se procede a realizar la asignación de cada cliente al almacén más cercano que esté abierto utilizando la distancia Euclidiana. Con esto, se obtiene la demanda total d_i correspondiente a cada almacén abierto en la idea n. Una vez que se tiene la demanda total de cada almacén i abierto se resuelve el nivel inferior asociado a cada uno de ellos utilizando el método de los multiplicadores de Lagrange, obteniendo así el valor de la cantidad óptima de pedido Q_i de cada almacén y el máximo de faltantes B_i.

Con la asignación de clientes a los almacenes y la solución del nivel inferior, es posible calcular el valor de la función objetivo f de cada idea n, la cual es tomada como indicador para ordenar las ideas de forma no decreciente en cada grupo m. Luego, se nombra como centro c, a la idea con la mejor función objetivo de cada grupo.

Una vez que se han agrupado las ideas, la siguiente fase del algoritmo de lluvia de ideas para el problema bi-nivel es la de generación de nuevas ideas, las cuales se forman aplicando los operadores de permutación y combinación, los cuales se describen en las siguientes subsecciones.

4.2.2 Operador de perturbación

El operador de pertubación se aplica a una sola idea. En este caso se utiliza un movimiento de intercambio, es decir, se evalúa el costo de cerrar (c_cerrar) alguno de los almacenes y se cierra el que aporte mayor costo a la función objetivo de la idea actual. Una vez cerrado el almacén de mayor costo, se incorpora a la solución el almacén que menor costo agregue a la función objetivo (c_abrir). Lo anterior se resume en el siguiente pseudocódigo.

```
   // Movimiento de perturbación()
   Data: idea, p, c_cerrar, c_abrir, max = −∞, min = ∞
 1 for i ≤ p do
 2     if idea[i] = 1 then
 3         if c_cerrar ≥ max then
 4             max = c_cerrar
 5             ind = i
 6         end
 7     end
 8 end
 9 idea[ind] = 0
10 for i ≤ p do
11     if idea[i] = 0 then
12         if c_abrir ≤ min then
13             min = c_abrir
14             ind = i
15         end
16     end
17 end
18 idea[ind] = 1
   Result: idea_new
```

Figura 4.5: Pseudocódigo para la perturbación

4.2.3 Combinación

Para el operador de combinación se requieren dos ideas $(idea_1, idea_2)$, dadas estas dos ideas se evalúa el costo de seleccionar una parte de ambas ideas para generar una nueva.

Para evaluar dicho valor, se crean todas las posibes combinaciones que se pueden hacer con p almacenes abiertos utilizando los almacenes abiertos de las dos ideas. Después de esto, de forma voraz se selecciona la solución con menor valor de la función objetivo.

// Movimiento de combinación()
Data: $idea_1,\ idea_2, min = \infty$
$i = 0$
$fo\_ideas = \emptyset$
$ideas\_nuevas = \emptyset$
1 Generar todas las posibles combinaciones de p almacenes abiertos con $idea_1$ e $idea_2$
2 **for** $i \leq |ideas\_nuevas|$ **do**
3 | Calcular función objetivo de cada idea generada
4 **end**
5 Se identifica la mejor idea generada $idea_{new}$
Result: $idea_{new}$

Figura 4.6: Pseudocódigo para la combinación

4.2.4 Actualización

Para la fase de actualización de la población de ideas, primero se calcula el valor de la función Hash de la nueva solución para saber a cual grupo debe pertenecer. Una vez que se realiza lo anterior, hay que calcular el valor objetivo de la nueva idea y compararlo con el valor objetivo de las demás ideas pertenecientes a dicho grupo. Si el valor de la función objetivo de la nueva idea es mejor que el valor objetivo de la peor idea, entonces se reemplaza la peor idea del grupo por la nueva y se ordenan de manera no creciente con respecto a las funciones objetivo.

Una vez descrito el algoritmo de lluvia de ideas, en la siguiente sección se comienza a describir el algoritmo genético propuesto para la comparación de soluciones del problema binivel.

4.3 Algoritmo genético para resolver el problema binivel bajo estudio

"En la naturaleza, los individuos de una población compiten entre sí en la búsqueda de recursos como comida, agua y refugio. También, los miembros de una misma especie suelen competir en la búsqueda de un compañero. Los individuos que tienen éxito en sobrevivir y en atraer compañeros, tienen mayor probabilidad de generar un gran número de descendientes. Por el contrario, los individuos que no son tan capaces, tienen un menor número de descendientes. Esto significa que los genes de los individuos mejor adaptados se heredarán en sucesivas generaciones a un número grande de individuos", Goldberg (1989). La combinación de buenas características que vienen de diferentes antecesores, puede a veces producir nuevos individuos, los cuales tienen una mayor adaptación a cualquiera de sus antecesores. De esta manera, las especies evolucionan logrando descendientes cada vez mejor adaptados en el entorno en el que viven.

Los Algoritmos Genéticos (GA) usan una analogía directa con el comportamiento natural. Estos algoritmos se basan en una población de *individuos*, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor o puntuación (función de adaptación), relacionado con la calidad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por algún recurso. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos, descendientes de los anteriores, los cuales comparten algunas de las características de sus padres. Cuanto menor sea la adaptación de un individuo, menor será la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones.

De esta manera, se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas características en comparación con la población anterior. Así, a lo largo de las generaciones las buenas características se propagan a través de la población. Favoreciendo el cruce de los individuos mejor adaptados, van siendo exploradas las áreas más prometedoras del espacio de búsqueda. "Si el GA ha sido bien diseñado, la población convergerá hacia una solución óptima o muy cerca del óptimo del problema", Uribe et al.

(2019).

Los algoritmos de evolución imitan al proceso natural, tales como selección, cruza, mutación, migración, localidad y vecindad. Al principio del algoritmo, un número de individuos son aleatoriamente inicializados. Entonces, la función objetivo es evaluada para este conjunto de individuos. La primera población (o incial) es producida.

Si el criterio de paro no ha sido satisfecho (alcanzar un máximo número de generaciones, generaciones sin mejora, etc.), una nueva generación comienza. Los individuos son seleccionados acorde a su función de adaptación para la producción de descendientes. Los padres son cruzados para producir dichos descendientes. Todos los descendientes serán mutados con una cierta probabilidad. Entonces, la función de adaptación de los descendientes es calculada. Los descendientes son insertados dentro de la población reemplazando a los padres, produciendo una nueva generación. El ciclo se realizará hasta que se alcance el criterio de paro.

En la Figura 4.7 se muestra el diagrama general de un algoritmo genético.

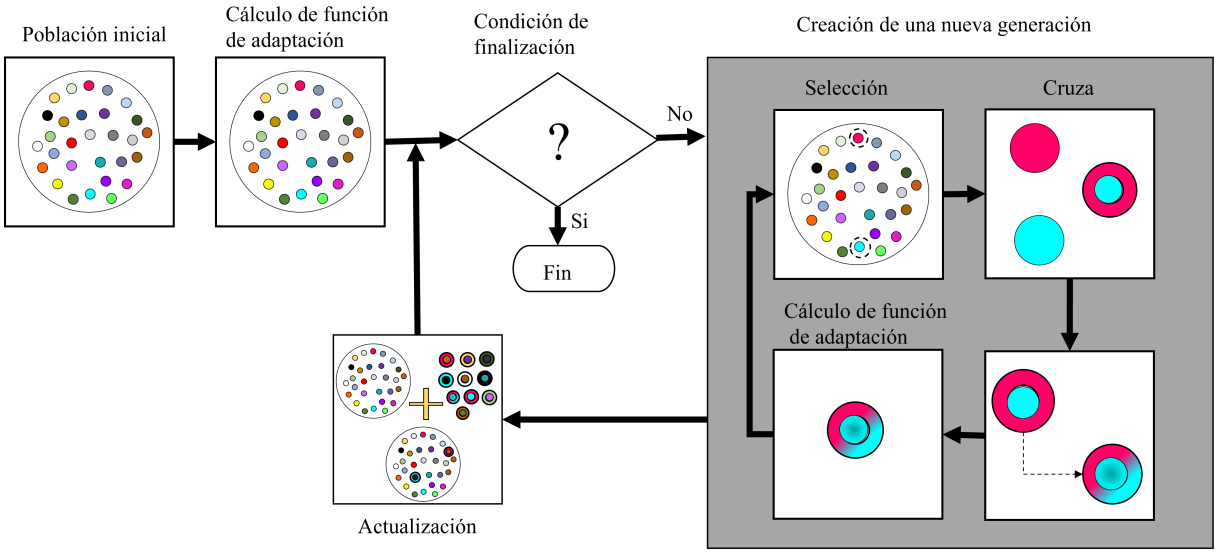


Figura 4.7: Diagrama de un algoritmo genético

Antes de pasar a la definición de cada una de los componentes principales del GA implementado para resolver el problema binivel, es necesario recalcar las notaciones de

las componentes del GA. La Tabla 4.2 presenta las diferentes notaciones utilizadas en el GA.

Tabla 4.2: Componentes del algoritmo genético

componente del GA	Modelo matemático
Individuo	Solución
Gen	Variable
Función de adaptación	Función objetivo

También, es importante mencionar que cada uno de nuestros individuos estará formado por un vector binario, cuyas componentes representan los genes como se muestra en la Figura 4.8.

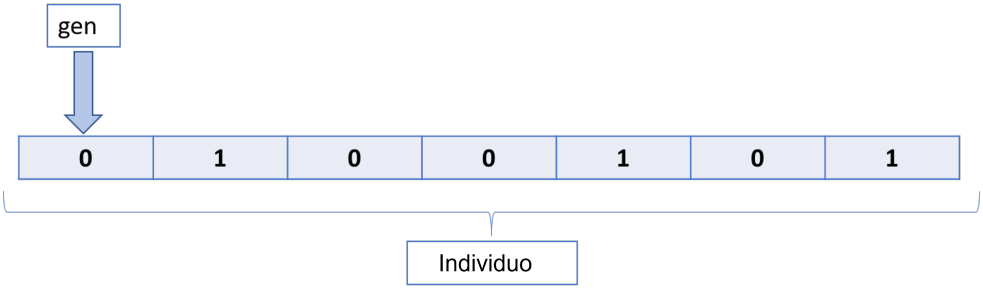


Figura 4.8: Representación de un individuo de la población

Después de mencionar las notaciones y la representación de un individuo, pasaremos a revisar cada una de las componentes del GA.

4.3.1 Generación inicial

Para comenzar, es necesario tener una población inicial de individuos, para esto se tomó una población totalmente aleatoria que sea factibles con respecto al número de almacenes abiertos y al número de sitios potenciales. Para cada uno de los individuos se calcula su función de adaptación. Como se mencionó anteriormente, la función de adaptación de cada uno de los individuos la representa el valor de la función objetivo del problema binivel. Recordemos que para obtener el valor de la función objetivo es necesario también resolver el nivel inferior y después evaluar dicha función objetivo.

4.3.2 Selección

La selección determina a cuales individuos elegir como padres para la fase de combinación y cuántos descendientes producen dichos padres.

Existen diversas técnicas de selección, entre las que se encuentran:

- Ruleta
- Muestreo estocástico universal
- Selección local
- Selección truncada
- Torneos

La técnica de selección utilizada en este trabajo fue la de por torneos, la cual consiste en escoger al azar un número de individuos de la población, fijar el número de individuos que se involucrarán en cada torneo, (con o sin reemplazamiento), seleccionar el mejor individuo de este grupo, y repetir el proceso hasta que el número de individuos seleccionados coincida con el tamaño de la población.

Para realizar los torneos, primero se ordenan a los individuos de mejor a peor con respecto al valor de la función de adaptación. Después, se toma al primer individuo y se enfrenta aleatoriamente con otro, si el valor de la función de adaptación del primero es mayor a la del otro individuo, se asigna un punto al primer individuo, de lo contrario, se asigna el punto al otro. Un torneo completo termina cuando el último individuo de la población se ha enfrentado a algún otro individuo. Después de un número predefinido de torneos, se suman los puntos de cada individuo, y los mejores serán utilizados en la fase de cruza, la cual se presenta en la próxima sección.

4.3.3 Cruza

Para esta fase se necesita tener dos padres. La mitad de los individuos que obtuvieron mayor número de torneos ganados en la fase de selección, son seleccionados para realizar el cruzamiento. Una vez ubicados los individuos que van a fungir como padres para la cruza, se van seleccionando uno por uno, y aleatoriamente se cruzan con otro

individuo de la población.

De cada uno de los padres seleccionados se toma aleatoriamente una parte de los genes de un padre y la mitad de los genes del otro (ver la Figura 4.9). Esto puede acarrear problemas al posiblemente formar nuevos individuos infactibles. Para estos casos, después del cruce se pasa a una fase de reparación, la cual se explica en la siguiente subsección.

```
   // Proceso de cruza()
   Data: individuo1, individuo2, individuo_new, p
 1 # p es el número de genes en cada individuo
 2 individuo_new → ∅

 3 for i < p do
 4 |   if i < p/2 then
 5 |   |   individuo_new[i] = individuo1[i]
 6 |   end
 7 |   else
 8 |   |   individuo_new[i] = individuo2[i]
 9 |   end
10 end
   Result: individuo_new
```

Figura 4.9: Pseudocódigo para la cruza

Una vez efectuados todos los cruces, con cierta probabilidad los nuevos individuos generados, pasarán a la fase de mutación.

4.3.4 Reparación

Cuando los nuevos individuos creados en la fase de cruza son infactibles se realiza una reparación, la cual consiste en lo siguiente: primero se revisa si el nuevo individuo cuenta con exceso de genes igual a 1 o al contrario contiene menos genes iguales a 1 de los que debe tener. En el primer caso, se evalúa el impacto en la función de adaptación del individuo al hacer cero cada uno de los genes que actualmente tienen valor 1, para después convertir en cero los genes necesarios que aporten mayor valor a la función de adaptación. Y de forma similar para el caso en donde se tienen menos genes iguales a 1 de los necesarios, se evalúa el impacto de asignar cada uno de los genes que estan en cero y se asignan los necesarios que generen mayor impacto positivo a la función de aptitud

del nuevo hijo (ver la Figura 4.10).

```
   // Reparación()
   Data: individuo, p, gen_1, adapt, max = −∞, min = ∞
 1 gen_1 número de genes igual a 1 que debe tener el individuo para ser factible.
 2 # p es el número de genes en el individuo
 3 cont = 0

 4 for i < p do
 5 |  if individuo[i] == 1 then
 6 |  |  cont_++
 7 |  end
 8 end
 9 while cont ≠ p do
10 |  if cont > gen1 then
11 |  |  for i ≤ p do
12 |  |  |  if individuo[i] = 1 then
13 |  |  |  |  if adapt[i] ≥ max then
14 |  |  |  |  |  max = adapt[i]
15 |  |  |  |  |  ind = i
16 |  |  |  |  end
17 |  |  |  end
18 |  |  end
19 |  |  individuo[ind] = 0
20 |  |  count−−
21 |  |  max = ∞
22 |  end
23 |  else
24 |  |  for i ≤ p do
25 |  |  |  if individuo[i] = 0 then
26 |  |  |  |  if adapt[i] ≤ min then
27 |  |  |  |  |  min = adapt[i]
28 |  |  |  |  |  ind = i
29 |  |  |  |  end
30 |  |  |  end
31 |  |  end
32 |  |  individuo[ind] = 1
33 |  |  count_++
34 |  |  min = −∞
35 |  end
36 end
   Result: individuo
```

Figura 4.10: Pseudocódigo para la reparación

4.3.5 Mutación

El proceso de mutación se lleva a cabo en una parte de los nuevos individuos generados en la fase anterior. Este proceso se lleva a cabo de manera similar a la perturbación en el algoritmo de lluvia de ideas. Para este caso, con cierta probabilidad se toma un individuo y se hace una modificación en los genes, intercambiando aleatoriamente un gen que es igual a cero a uno y viceversa, un gen que es igual a 1 se cambia a cero. El siguiente pseudocódigo mostrado en la Figura 4.11 resume el proceso anterior.

```
// Proceso de mutación()
Data: individuo, p, gen1, gen2, Flag = 0
# p es el número de genes en el individuo

gen1 = aleatorio(0, p)
if individuo[gen1] = 1 then
    while Flag == 0 do
        individuo[gen1] = 0
        gen2 = aleatorio(0, p)
        if gen1 ≠ gen2 & gen2 == 0 then
            individuo[gen2] = 1
            Flag = 1
        end
    end
end
else
    idea[gen1] = 1
    while Flag == 0 do
        gen2 = aleatorio(0, p)
        if gen1 ≠ gen2 & gen2 == 1 then
            individuo[gen2] = 0
            Flag = 1
        end
    end
end
Result: individuo
```

Figura 4.11: Pseudocódigo para la mutación

4.3.6 Actualización de población

Una vez concluidas las fases de cruzamiento y mutación, se pasa a la actualización de la población, la cual será de forma elitista. Es decir, se conservarán a los mejores individuos entre la generación anterior y los hijos de la nueva generación, cuidando siempre conservar el mismo número de individuos.

4.4 Algoritmo exhaustivo para resolver el problema binivel

Muchos problemas de optimización sólo pueden resolverse de manera exacta explorando todas las posibles soluciones factibles del problema y eligiendo aquella que sea la mejor. Lo anterior también sucede en los problemas de programación binivel discretos. Debido a esto, los algoritmos exhaustivos pueden ser utilizados para resolver este tipo de problemas, pero en instancias pequeñas, ya que al ser problemas combinatorios, entre más grande sea el problema, más demandante será en esfuerzo computacional. Inclusive, puede volverse imposible de resolver un problema de tamaño mediano o grande por una computadora, debido al el excesivo número de operaciones que se deben hacer.

Para el caso del problema presentado en este trabajo de tesis, como se ha mencionado en secciones anteriores, no hay resultados previos que nos ayuden a comparar la calidad de nuestras soluciones, por lo que un algoritmo exhaustivo puede ser de gran ayuda para resolver casos pequeños y tener los resultados como base. En la sección de experimentación computacional se mostrarán los casos para los cuales fue posible llegar a una solución óptima con este algoritmo. De todos modos, en esta sección nos enfocaremos solo en describirlo.

El algoritmo exhaustivo es sencillo de describir, ya que lo único que se realiza es de forma ordenada ir encontrando todas las posibles configuraciones de soluciones factibles al problema binivel (soluciones para el líder). Hay que hacer notar que la forma de representar las soluciones en el algoritmo exhaustivo, es la misma utilizada en los dos algoritmos anteriores. Una solución está formada por un vector binario con tantos unos como almacenes se deban abrir en el problema. Entonces, el encontrar todas las posibles soluciones se traduce en encontrar todas las posibles permutaciones de m en n donde m es el número de sitios potenciales y n el número de almacenes por abrir. Cada solución factible que se encuentre, es evaluada de acuerdo a la función objetivo del problema (recordando que para cada solución hay que resolver el problema del nivel inferior) y compararla con la mejor solución actual; si la mejora, se reemplaza pasando a ser la mejor solución, si no, se deshecha.

El pseudocódigo del algoritmo exhaustivo se presenta a continuación.

```
   // Algoritmo exhaustivo
   Data: m, n, min = ∞, total_permutaciones
 1 sol = ∅
 2 sol_best = ∅
 3 OF = 0
 4 permut[i] función que da cada una de las permutaciones de m en n de forma ordenada

 5 for i < total_permutaciones do
 6 |   sol = permut[i]
 7 |   OF = funcion_objetivo(sol)
 8 |   if OF < min then
 9 |   |   min = OF sol_best = sol
10 |   end
11 end
   Result: sol_best, OF
```

Figura 4.12: Pseudocódigo del algoritmo exhaustivo

Capítulo 5

Experimentación

En esta sección se describen los resultados numéricos obtenidos de la experiemtación computacional. Toda la experimentación fue realizada utilizando una computadora de escritorio con procesador de 3.6 GHz Intel Core i7-4790 con 32 GB de RAM en Windows 8.1 Pro. El algoritmo fue implementado en Visual Studio Express 2012. Al realizar la revisión de literatura no se encontraron trabajos previos que estudien un problema binivel de localización de la forma en que se propone en este trabajo. Por lo tanto, se presentan resultados computacionales del modelo propuesto, considerando distintos conjuntos de instancias adaptadas, las cuales se describen a continuación.

5.1 Definición de instancias

Las instancias usadas para las pruebas fueron obtenidas de la librería de Beasley's OR [1]; en particular, para el problema de localización de almacenes sin capacidades, que son las mismas que fueron utilizadas en Camacho-Vallejo et al. (2014). Éstas instancias fueron adaptadas para incluir la parte asociada con la decisión de inventarios para completar los parámetros necesarios para el problema binivel. Para la parte de inventarios, se generaron valores aleatorios para cada parámetro distribuídos en intervalos predefinidos. La tabla 5.1 muestra el rango para los valores de cada parámetro.

[1] http://people.brunel.ac.uk/ mastjjb/jeb/orlib/files/

Tabla 5.1: Paámetros utilizados para generar los datos faltantes de las instancias de prueba.

	pequeñas	medianas	grandes
clientes	50	50	1000
almacenes	[16,25]	50	100
p	[3,7]	[5,10]	[5,20]
costo de apertura	[400, 1500]	[400, 1500]	[400, 1500]
costo de inventario	[10, 300]	[10, 300]	[10, 300]
costo de ordenes atrasadas	[0, 8]	[0, 8]	[0, 8]
costo por ordenas atrasadas por unidad de tiempo	[5, 200]	[5, 200]	[5, 200]
número máximo de órdenes atrasadas	[0, 100]	[0, 100]	[0, 100]
costo de poner una orden	[80, 2650]	[80, 2650]	[80, 2650]

En total, se crearon 37 instancias para tener nuestro conjunto de prueba. Este conjunto se divide en 15 instancias pequeñas, 10 medianas y 12 grandes. Para cada instancia, todos los par parámetros se fijan con un valor seleccionado de los intervalos descritos anteriormente; es decir, se consideran instancias deterministas.

5.2 Afinación de parámetros de los algoritmos

El objetivo de esta parte del trabajo es evaluar la influencia de los parámetros correspondientes a cada uno de los dos algoritmos propuestos (BSO y GA), lo anterior con la finalidad de obtener los mejores valores de cada parámetro para usar en la experimentación.

En primer lugar, analizaremos el caso del algoritmo de lluvia de ideas. Para esto se tomaron en cuenta 6 parámetros (factores) y varios valores predeterminados (niveles) en un diseño de experimentos factorial de $2 \times 3^4 \times 2$. Los factores y niveles son: tamaño de población inicial (ip=1000 o 1500), número de iteraciones BSO (it=500, 1000, o 2500), probabilidad de combinación (combinacion=0.25, 0.5, o 0.6), probabilidad de combinar dos centros (combinación c-c=0.3, 0.5, o 0.8), probabilidad de combinar un centro y una idea (combinación c-i=0.2, 0.5, o 0.7) y probabilidad de perturbar un centro (perturbación-c=0.5 o 0.7). Cada combinación de niveles de los factores da como resultado una posible configuración del algoritmo. Como se mencionó anteriormente, se desea evaluar la influencia de los factores en el desempeño del algoritmo; para esto, se seleccionaron 12 ins-

tancias de las 37 totales por medio de un muestreo estratificado aleatorio para realizar las pruebas, donde cada tamaño de instancia fue considerado como un estrato y de cada uno de ellos se seleccionaron de forma aleatoria 4 instancias. Debido a la aleatoriedad existente en el algoritmo propuesto, cada una de las configuraciones fue corrida 5 veces (réplicas).

Para saber cual configuración es la mejor para este algoritmo, primero hay que definir cuando se considera un "éxito", es decir, que hay un aporte positivo de dicha configuración a la función objetivo. Lo anterior se logra definiendo f y $f_{promedio}$ como el mejor valor y el valor promedio de la función objetivo obtenida en las 5 réplicas de cada una de las configuraciones de parámetros, respectivamente. Entonces, como el objetivo del problema en cuestión es de minimización, se considera un éxito si $f - f_{promedio} < 0$.

Los resultados del análisis de varianza aplicado a las proporciones de éxito indican que el tamaño de población inicial, el número de iteraciones y el uso de combinación son los factores significantes siendo su valor de p cercano a 0. También se puede observar que la interacción del tamaño de población inicial y la probabilidad de combinación es significativa.

Las Figuras 5.1 y 5.2 muestran los efectos principales para la proporción de éxitos y para la interacción entre ellos, respectivamente. De este análisis se puede concluir que una buena configuración de parámetros del BSO consiste en utilizar una población inicial de ideas de tamaño 1500, con un número total de 2500 iteraciones. Para la parte de las probabilidades se debe utilizar una probabilidad de combinación de 0.60, y por consecuencia una probabilidad de 0.40 para la perturbación de ideas. La probabilidad de combinar dos centros se fija en 0.60 y la de centro-idea en 0.30. Por último, la probablidad de perturbar una idea se fija en 0.50.

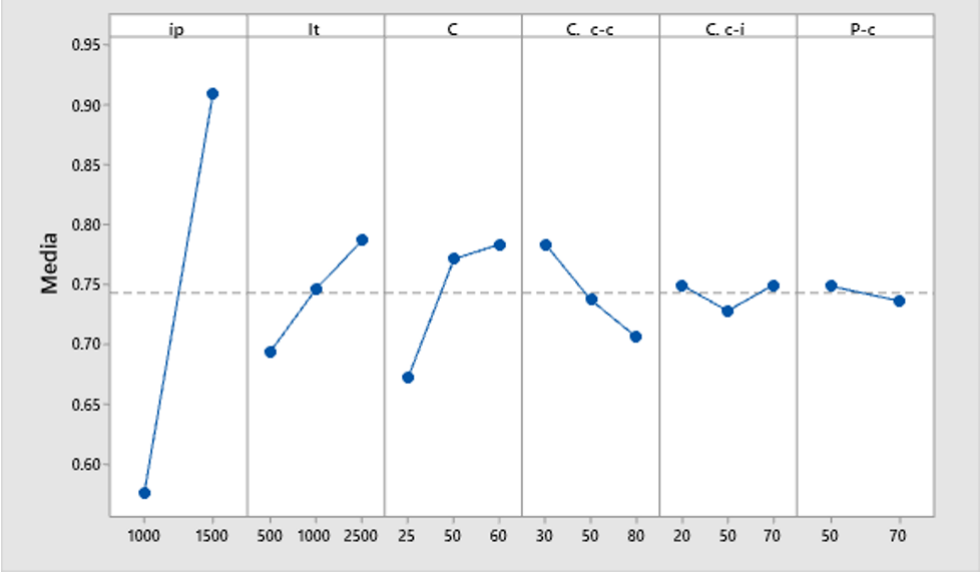


Figura 5.1: Gráfico de los efectos principales para la proporción de éxitos del BSO

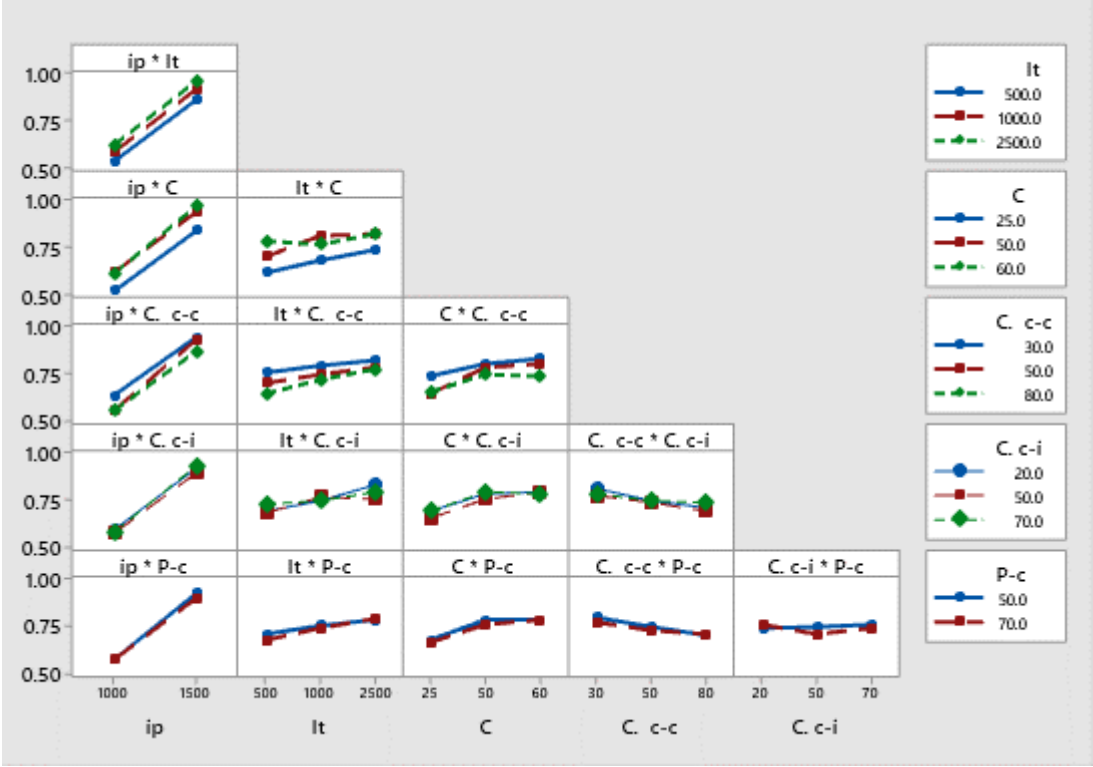


Figura 5.2: Gráfico de interacción para las proporciones de éxitos para el BSO

De forma similar, se realizó un análisis para encontrar una configuración adecuada para los parámetros del algoritmo genético. Para la parte del diseño de experimentos se

utilizaron 4 factores. Comenzando con el factor correspondiente a la población inicial, el número de torneos y el número de generaciones se tomaron 3 niveles (ip= $1000, 1200, 1500$, torneos= $3, 4, 5$, generaciones= $50, 100, 150$), mientras que para la probabilidad de muta se utilizaron 4 niveles (muta= $.1, .2, .3, .4$). Con lo anterior, se realizó un diseño factorial de $3^3 \times 4$. Se utilizaron las mismas instancias seleccionadas para la afinación de los parámetros del BSO y también se realizaron las 5 corridas de cada instancia.

Del análisis anterior para el GA, se obtuvo que los valores a utilizar en la experimentación deben ser: población inicial 1200, 5 torneos, probabilidad de permutación .2 y 150 generaciones (ver Figuras 5.3 y 5.4).

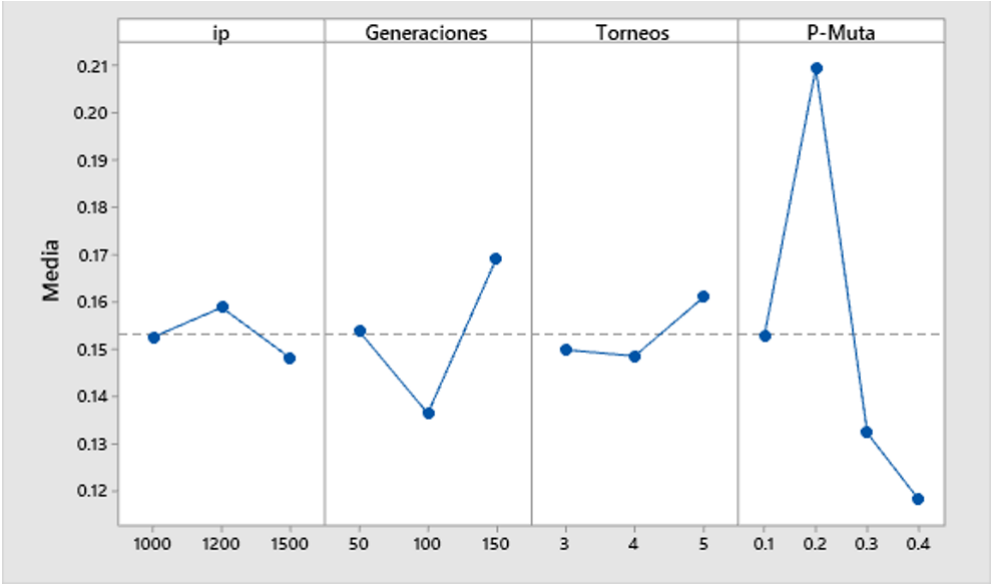


Figura 5.3: Gráfico de los efectos principales para la proporción de éxitos del GA

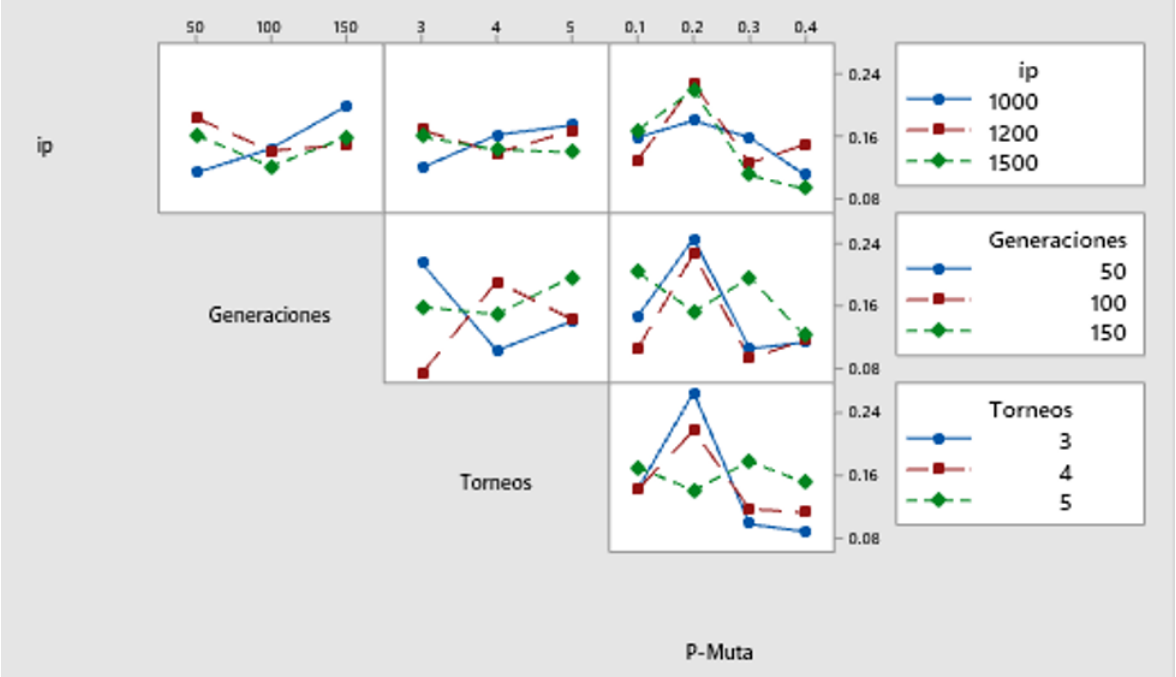


Figura 5.4: Gráfico de interacción para las proporciones de éxitos para el GA

Por último, es importante recordar que el algoritmo exhaustivo no involucra ningún parámetro ya que no importa el tamaño de la instancia, el algoritmo siempre funciona de la misma forma para cada corrida.

5.3 Resultados obtenidos de la experimentación computacional

En esta sección se presentan los resultados obtenidos de la experimentación realizada con los algoritmos propuestos utilizando la configuración de parámetros obtenidos en la subección previa y los resultados obtenidos por el algoritmo exhaustivo para fines comparativos. Además, se presentan algunas conclusiones derivadas de los resultados numéricos, seguidas por un pequeño análisis estadístico.

Debido a la naturaleza estocástica del BSO, se llevaron a cabo 30 corridas para cada una de las 37 instancias. Es importante notar que para cada solución del nivel superior, se resuelve de manera independiente a optimalidad el problema del nivel inferior utilzando el método de multiplicadores de Lagrange (ver Apéndice 5.5), lo anterior es necesario para obtener soluciones binivel factibles. El hecho de tener que resolver a optimalidad el nivel

inferior, incurre en el incremento de tiempos computacionales conforme el tamaño del problema crece, sin embargo crece de forma controlada. Este comportamiento es común en algoritmos que deben resolver problemas anidados como lo son los problemas binivel Dempe (2002).

Los resultados obtenidos se pueden observar en las Tablas 5.2 y 5.3, donde la columna "Instancia" corresponde al nombre de la instancia de prueba. Las siguientes dos columnas, f_exh y t corresponden al valor de la función objetivo y el tiempo total requerido por el algoritmo exhaustivo para resolver la instancia a optimalidad. En los casos donde la solución óptima no fue alcanzada por el algoritmo exhaustivo debido a las limitaciones computacionales, el mejor valor de la función objetivo alcanzado en un tiempo de 6 horas es reportado como "el mejor valor conocido" (marcado con $*$). Luego, para el BSO y el GA, se presentan las columnas f_bst, f_avg, and f_wst. Cada una de ellas reporta el mejor, promedio y peor valor de la función objetivo obtenida por cada algoritmo en las 30 corridas, respectivamente. Es importante notar que cuando aparece "-" en la tabla, indica que se llegó al óptimo. Por último, la columna t_avg muestra el tiempo promedio requerido (en segundos) para cada una de las 30 corridas de cada instancia.

Tabla 5.2: Resultados obtenidos con los algoritmos de prueba para las instancias pequeñas.

	Algoritmo exhaustivo		Algoritmo BSO				GA			
Instancia	f_exh	t (s)	f_bst	f_avg	f_wst	t_avg (s)	f_bst	f_avg	f_wst	t_avg (s)
small1	3,356,130	0.62	-	-	-	3.85	-	-	-	5.07
small2	2,963,010	0.27	-	-	-	2.88	-	-	-	2.71
small3	3,131,050	0.29	-	-	-	3.21	-	-	-	4.13
small4	3,257,880	0.06	-	-	-	6.83	-	-	-	5.14
small5	2,000,340	0.33	-	-	-	5.57	-	-	-	12.17
small6	3,650,760	21.93	-	-	-	7.39	-	-	-	11.31
small7	3,646,660	2.44	-	-	-	3.77	-	-	-	5.12
small8	2,494,450	7.93	-	-	-	6.40	-	-	-	9.45
small9	2,550,890	2.51	-	-	-	5.51	-	-	-	11.64
small10	2,694,560	0.74	-	-	-	5.00	-	-	-	8.35
small11	2,231,010	8.13	-	-	-	6.45	-	-	-	13.40
small12	2,650,250	8.10	-	-	-	4.46	-	-	-	8.99
small13	1,819,900	2.50	-	-	-	5.95	-	-	-	8.89
small14	3,171,700	158.25	-	-	-	9.43	-	-	-	15.43
small15	4,063,960	50.89	-	-	-	7.82	-	-	-	8.75

Tabla 5.3: Resultados obtenidos con los algoritmos de prueba para las instancias medianas y grandes.

	Algoritmo exhaustivo		Algoritmo BSO				GA			
Instancia	f_exh	t (s)	f_bst	f_avg	f_wst	t_avg (s)	f_bst	f_avg	f_wst	t_avg (s)
medium1	3,051,480*	21,600.00	**2,489,420**	2,749,410.00	3,573,500	97.63	**2,489,420**	3,085,330	3,681,240	93.75
medium2	2,318,980	7,532.60	-	2,413,498.89	2,566,340	77.60	-	1,336,310	2,672,620	66.06
medium3	1,819,010	1,183.94	-	1,861,306.67	1,959,950	68.78	-	-	-	59.09
medium4	2,425,750	160.39	-	-	-	60.17	-	-	-	61.62
medium5	2,041,190*	21,600.00	**1,940,230**	1,992,877.65	2,140,760	32.37	**1,940,230**	2,075,247	2,210,263	52.32
medium6	3,460,120*	21,600.00	**3,239,410**	3,280,429.29	3,360,390	30.10	**3,239,410**	3,311,975	3,384,540	54.57
medium7	1,783,870	7,529.41	-	1,796,631.11	1,826,480	68.67	-	-	-	75.00
medium8	1,741,880	159.55	-	-	-	47.41	-	-	-	49.26
medium9	1,716,260	1,210.54	-	-	-	51.29	-	-	-	53.92
medium10	2,002,790	7,669.98	-	-	-	59.33	-	-	-	70.50
large1	17,255,700*	21,600.00	**16,057,000**	16,057,000	16,057,000	1,320.16	**16,057,000**	16,057,000	16,057,000	1,407.02
large2	53,881,200*	21,600.00	**29,749,100**	29,749,100.00	29,749,100	1,325.27	**29,749,100**	29,762,080	29,814,000	1,553.55
large3	33,168,600*	21,600.00	**16,057,000**	16,057,000.00	16,057,000	1,486.26	**16,057,000**	16,057,000	16,057,000	1,697.26
large4	45,103,900*	21,600.00	**28,403,800**	28,403,800.00	28,403,800	1,525.41	**28,403,800**	28,403,800	28,403,800	1,696.04
large5	53,475,500*	21,600.00	**35,842,000**	35,842,000.00	35,842,000	1,462.50	**35,842,000**	35,911,240	36,188,200	1,656.52
large6	39,851,600*	21,600.00	**21,413,700**	21,413,700.00	21,413,700	1,367.00	**21,413,700**	21,413,700	21,413,700	1,565.32
large7	28,789,200*	21,600.00	**18,723,100**	18,723,100.00	18,723,100	1,467.76	**18,723,100**	18,723,100	18,723,100	1,567.91
large8	29,946,200*	21,600.00	**13,829,500**	13,847,133.33	13,858,600	1,625.16	**13,829,500**	13,829,500	13,829,500	1,649.09
large9	52,356,400*	21,600.00	**22,989,500**	22,989,500.00	22,989,500	1,648.42	**22,989,500**	22,989,500	22,989,500	1,697.60
large10	33,463,300*	21,600.00	**20,866,800**	20,866,800.00	20,866,800	1,524.55	23,057,800	23,198,240	23,598,400	1,751.26
large11	40,165,800*	21,600.00	**23,775,200**	23,793,050.00	23,795,600	1,485.21	**23,775,200**	23,970,840	24,305,400	1,758.14
large12	26,440,500*	21,600.00	**17,036,800.00**	17,120,000.00	17,203,200	1,548.23	17,077,000	17,200,750	17,324,500	1,751.83

Como puede observarse en la Tabla 5.2, las primeras 15 instancias corresponden a las instancias de tamaño pequeño. En todas las instancias se alcanzó el óptimo por los tres algoritmos. Más aún, el óptimo se alcanza por el algoritmo de lluvia de ideas y el algoritmo genético en las 30 corridas de cada instancia. Aunque hay solo 6 de las 15 instancias en las que el tiempo computacional del algoritmo BSO es menor al tiempo requerido por el exhaustivo, hay 13 de las 15 instancias en las que el algoritmo de lluvia de ideas requirió menor tiempo computacional que el algoritmo genético para alcanzar el óptimo.

Para las instancias medianas (ver la Tabla 5.3), el tiempo computacional requerido se incrementa pero de forma controlada. Dicho tiempo varía entre 30 y 98 segundos, consumiendo menos tiempo que el algoritmo exhaustivo en todas las instancias; y comparado con el algoritmo genético, éste requiere menos tiempo en **7** de las **10** instancias. Con respecto al valor de la función objetivo, puede verse que el algoritmo de lluvia de ideas obtiene el óptimo en **7** de las **10** instancias, mientras que para las otras 3 instancias en las cuales el algoritmo exhaustivo no alcanza el óptimo durante el tiempo límite, el algoritmo de lluvia de ideas obtiene el "mejor valor conocido" (también obtenido por el algoritmo genético) en menor tiempo para las instancias medium5 y medium6.

En el caso de las instancias grandes (ver también en la Tabla 5.3), el algoritmo exhaustivo no resuelve a optimalidad las instancias. Esto es claro debido a la naturaleza combinatoria del problema. Para realizar una comparación justa, los resultados del algoritmo exhaustivo no se utilizan para evaluar el comportamiento del algoritmo de lluvia de ideas. En lugar de esto, se utilizó el algoritmo de lluvia de ideas comparado con los resultados del algoritmo genético. Con respecto al algoritmo de lluvia de ideas, éste obtiene "mejor valor conocido" en todas las 30 corridas en 9 de las 12 instancias, mientras que el algoritmo genético lo alcanza en solo 7 instancias. Además, el algoritmo de lluvia de ideas obtiene valores de función objetivo más bajos que el algoritmo genético en 2 de las 12 instancias (large10 y large12). En las 10 instancias restantes, se alcanza el mismo valor objetivo en ambos algoritmos. Para el algoritmo de lluvia de ideas, el tiempo computacional requerido para resolver las instancias grandes se incrementa de 22 a 26 minutos, aproximadamente. Mientras que por otra parte, el tiempo requerido por el algoritmo genético en todas las instancias es mayor al algoritmo de lluvia de ideas, llegando a tomar más de 55 minutos en algunas instancias.

Para mostrar de forma más clara los resultados obtenidos por el algoritmo de lluvia de ideas, en la tabla 5.4, se muestran los gaps con respecto al óptimo o al "mejor valor conocido". La primera columna muestra el mejor valor conocido de la función objetivo,

después, las siguientes dos columnas muestran el gap del algoritmo de lluvia de ideas con respecto al valor obtenido por el algoritmo exhaustivo. Los gaps *gap_bst* y *gap_avg* son calculados entre el mejor valor obtenido por el algoritmo exhaustivo y el mejor valor alcanzado por el algoritmo de lluvia de ideas, o el valor promedio de las 30 corridas de cada instancia, respectivamente. De forma similar, las dos últimas columnas muestran las holguras (*gaps*) entre los resultados obtenidos por el algoritmo exhaustivo y el algoritmo genético. La fórmula utilizada para estos cálculos es:

$$gap = \frac{f\_obtenido - f\_optimo}{f\_optimo} \times 100$$

Tabla 5.4: Holguras calculadas entre los algoritmos propuestos para las instancias medianas y grandes.

	Mejor valor	BSO		GA	
	obtenido	gap_bst	gap_avg	gap_bst	gap_avg
medium1	2,489,420	-18.419	-9.899	-18.419	1.109
medium2	2,318,980	0.000	4.076	0.000	7.625
medium3	1,819,010	0.000	2.325	0.000	0.000
medium4	2,425,750	0.000	0.000	0.000	0.000
medium5	1,940,230	-4.946	-2.367	-4.946	1.668
medium6	3,239,410	-6.379	-5.193	-6.379	-4.281
medium7	1,783,870	0.000	0.715	0.000	0.000
medium8	1,741,880	0.000	0.000	0.000	0.000
medium9	1,716,260	0.000	0.000	0.000	0.000
medium10	2,002,790	0.000	0.000	0.000	0.000
large1	16,057,000	-6.947	-6.947	-6.947	-6.947
large2	29,749,100	-44.788	-44.788	-44.788	-44.764
large3	16,057,000	-51.590	-51.590	-51.590	-51.590
large4	28,403,800	-37.026	-37.026	-37.026	-37.026
large5	35,842,000	-32.975	-32.975	-32.975	-32.845
large6	21,413,700	-46.266	-46.266	-46.266	-46.266
large7	18,723,100	-34.965	-34.965	-34.965	-34.965
large8	13,829,500	-53.819	-53.760	-53.819	-53.819
large9	22,989,500	-56.090	-56.090	-56.090	-56.090
large10	20,866,800	-37.643	-37.643	-31.095	-30.676
large11	23,775,200	-40.807	-40.763	-40.807	-40.320
large12	17,036,800	-35.566	-35.251	-35.413	-34.945

Es importante resaltar que, basados en la forma en la cual son calculados las holguras, un valor negativo indica que el resultado obtenido por el algoritmo de lluvia de ideas o el algoritmo genético son mejores que el "mejor valor conocido" obtenido por el algoritmo exhaustivo. Así, podemos notar en la Tabla 5.4, que para las tres instan-

cias medianas en las cuales el algoritmo exhaustivo no alcanza el óptimo, ambos, tanto el algoritmo de lluvia de ideas como el algoritmo genético mejoran el "mejor valor conocido" obtenido por el algoritmo exhaustivo en el tiempo límite. Además, las holguras correspondientes a las instancias grandes son negativos para ambos metaheurísticos. En 10 de las 12 instacias grandes, el algoritmo de lluvia de ideas y el algoritmo genético obtienen el "mejor valor conocido". En las últimas dos intancias, el algoritmo de lluvia de ideas presenta mejores holguras, mostrando estabilidad y un comportamiento eficiente.

Finalmente, las Figuras 5.5 y 5.6 muestran el comportamiento de los tiempos de ejecución (promedio) para ambos metaheurísticos implementados correspondientes a las instancias medianas y grandes, respectivamente. Puede observarse que para las instancias medianas, el algoritmo de lluvia de ideas muestra mejores resultados que el algoritmo genético en 7 de las 10 instancias. Aun así, no hay una gran diferencia entre los tiempos. Por otro lado, para las instancias grandes, hay una evidente diferencia entre los tiempos. Empíricamente, puede mencionarse que el algoritmo de lluvia de ideas reporta menor tiempo significativo que el algoritmo genético. Ésto último refleja las ventajas de usar el algortimo de lluvia de ideas para resolver el problema estudiado en esta tesis.

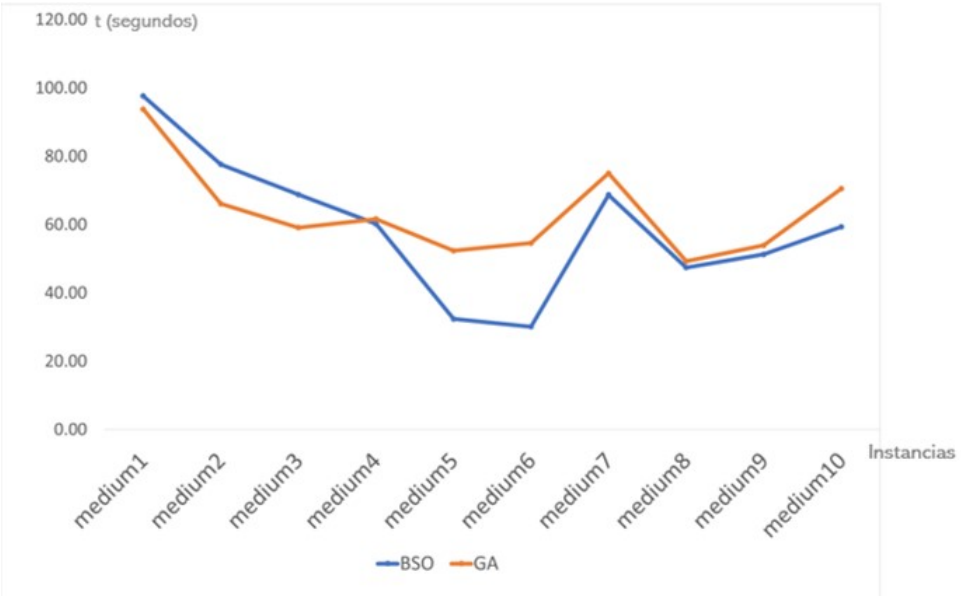


Figura 5.5: Tiempos computacionales en segundos requeridos por las metaherísticas implementadas para las instancias medianas.

Figura 5.6: Tiempos computacionales requeridos por las metaherísticas implementadas para las instancias grandes.

5.4 Análisis estadístico

Como se pudo observar en las Tablas 5.2 y 5.3, los resultados obtenidos por el algoritmo de lluvia de ideas y el algoritmo genético son muy parecidos en cuanto al mejor valor de la función objetivo alcanzada por ambos. Una forma de comparar la calidad de las soluciones es revisando el valor promedio de la función objetivo obtenida para cada instancia en las 30 corridas. Otra de las caracaterísticas para evaluar la eficiencia de un algoritmo es el tiempo de cómputo. Tomando en cuenta lo anterior, el análisis estadístico realizado se llevó a cabo de acuerdo a las dos métricas (valor promedio de la función objetivo y tiempo de cómputo).

Utilizando Minitab, se llevo a cabo un análisis individual, tanto para los tiempos de cómputo de ambos algoritmos como para el valor promedio de la función objetivo. La Tabla 5.5 muestra un resumen de la media, varianza y desviación estándar correspondientes a cada uno de los factores tomados en cuenta.

	f_avg	*t_avg (s)*
GA media	9,017,762.4732	926.7094
BSO media	8,933,123.4301	835.4203
GA Varianza	10.035E+13	656,187.8681
BSO Varianza	9.853E+13	531,761.9457

Tabla 5.5: Análisis estadístico para la comparación de los algoritmos

Por último, como parte del análisis estadístico, se realizó una prueba de hipótesis para datos pareados para probar si existe diferencia entre los tiempos de cómputo o los valores promedio de las funciones objetivo de ambos algoritmos. Con un nivel de confianza del 95 % se encontró que hay diferencia en las medias solamente de los tiempos, siendo el algoritmo de lluvia de ideas más eficiente que el algoritmo genético.

Capítulo 6

Conclusiones y trabajo futuro

En este trabajo de tesis se introdujo un problema de localización de almacenes considerando una política de inventario. El objetivo principal del problema es seleccionar en cuales sitios abrir almacenes de un grupo de sitios potenciales donde pueden ser localizados, lo anterior minimizando los costos de localización y de asignación de clientes a los almacenes. Una vez realizada la apertura de los almacenes y ya asignados los clientes a dichos almacenes abiertos, cada uno de los almacenes tienen una decisión que tomar, y esta es, la cantidad de pedido a realizar. Cada uno de los almacenes tienen restricciones físicas y operacionales tales como la máxima capacidad de cada almacén y el máxmo nivel de órdenes atrasadas permitidas. Aunque cada uno de los almacenes toma sus propias decisiones, ésto afecta a los costos tomados en cuenta en la decisión de localizacón. Debido a la evidente jerarquía presente en este trabajo se propuso un modelo binivel que fue el que mejor se acopló a la situación bajo estudio.

Para resolver el problema binivel resultante, se propusieron un algoritmo de lluvia de ideas y un algoritmo genético. Como se mencionó antes, el algoritmo de lluvia de ideas es un algoritmo que se ha introducido recientemente en la literatura y ha sido usado principalmente para problemas continuos, por lo tanto, su uso para resolver problemas discretos es escaso. Aunado a eso, se resolverá un problema de programación binivel. Es por esto, que el algoritmo de lluvia de ideas es considerado como una opción innovadora para resolver el problema. Por otro lado, el algoritmo genético es mas común verlo en la solución de problemas discretos. Ambos algoritmos son metaheurísticos poblacionales y fueron utilizados debido a su similitud en algunas componentes. Para el algoritmo de lluvia de ideas, la base es el agrupamiento de ideas (soluciones) y la creación de nuevas ideas por medio de los operadores de combinación y perturbación; mientras que el algoritmo genético se basa en la creación de una población inicial de individuos (soluciones) las cuales se van mejorando mediante los movimientos de cruzamiento y mutación de los individuos ya existentes.

Debido a la no linealidad del problema del nivel inferior, éste es resuelto usando el método de multiplicadores de Lagrange para cada decisión del nivel superior. Es importante mencionar que tanto el algoritmo de lluvia de ideas como el algoritmo genético consideran diversos parámetros, y por lo tanto, se tuvo que realizar una calibración para obtenerlos. Lo anterior, se logró realizando un diseño de experimentos. Con eso, se encontró una configuración de parámetros adecuada para llevar a cabo la experimentación computacional.

Además de los dos metaheurísticos propuestos, se implementó un algoritmo exhaustivo, el cual sirvió para comparar los resultados de las instancias pequeñas con el valor óptimo. Cabe mencionar que dicho algoritmo exhaustivo solo fue utilizado para resolver instancias pequeñas.

De los resultados obtenidos de la experimentación computacional, puede concluirse que el algoritmo propuesto de lluvia de ideas demostró un buen desempeño. En las 37 instancias, el algoritmo de lluvia de ideas obtiene el óptimo o el mejor valor conocido. Más aún, en 34 de las 37 instancias, el mejor valor fue obtenido en las 30 corridas de cada instancia. También, la holgura con respecto al óptimo es cero para las instancias pequeñas y en la mayoría de as instancias medianas; mientras que, para las instancias grandes, la diferencia entre las holguras no excedió 0.24 %. Esto confirma la estabilidad del algoritmo de lluvia de ideas. Con respecto a los tiempos computacionales, son relativamente bajos (menos de 100 segundos) para las instancias pequeñas y medianas, mientras que para las instancias grandes mantiene un tiempo razonable debido a la naturaleza estratégica del problema. Para el algoritmo genético se observó un comportamiento similar al algoritmo de lluvia de ideas en cuanto a valor de funciones objetivo. Pero en lo que respecta a los tiempos computacionales el algoritmo de lluvia de ideas dió indicios de ser más rápido. Debido a esto, en la sección de análisis estadístico se realizaron algunas pruebas no paramétricas para revisar si la diferencia entre los tiempos fue significativa. Los resultados de la prueba de hipótesis confirmaron que el algoritmo de lluvia de ideas resultó ser mejor al algoritmo genético en tiempos computacionales, por lo que el algoritmo de lluvia de ideas es una buena opción para resolver el problema estudiado en esta tesis. Más aun, servirá como punto de referencia para futuras investigaciones en el área.

6.1 Trabajo futuro

El trabajo presentado en esta tesis, puede ser extendido para que se ajuste mejor a diferentes aplicaciones realizando algunas adecuaciones pertinentes. Algunas posibles formas de extender esta investigación se presentan a continuación.

Una primer variante del problema podría ser el considerar el caso estocástico, es decir que las demandas de los clientes o las capacidades de los almacenes fueran variables. Este tipo de problemas se acercan más a resolver problemas de la realidad en donde los clientes no siempre realizan la misma cantidad de pedido. Tomando en cuenta la parte estocástica del problema, el nivel inferior pasaría a ser más complejo de resolver, por lo que se requeriría otro método para resolver el nivel inferior.

Otra adecuación que podría hacerse al problema propuesto en este trabajo de tesis es la de considerar el caso en el cual los clientes demandan más de un tipo de producto. Esto haría que los almacenes tuvieran que realizar más de un pedido en cada periodo. Lo anterior no afectaría al nivel superior si no colocamos capacidades a los almacenes, solo modificaría el nivel inferior en el que cada almacén debe realizar más de una orden. En caso de considerarlo multiproducto sería más problemático a la hora de resolver el nivel inferior.

Una última propuesta para trabajo futuro podría ser que demos a los clientes la posibilidad de ser servidos por más de un almacén (también en este caso podrían considerarse diferentes productos). Se consideraría una capacidad de servicio a cada almacén y consideraría la posibilidad de que surtieran cargas parciales a cada cliente. Con lo anterior y agregando la parte estocástica podría tenerse un acercamiento a los problemas que se dan en empresas dedicadas al comercio online cuyos clientes demandan pedidos de diferentes servidores. Este tipo de problemas sería el más complicado de atender, ya que para su solución tanto el nivel superior como el inferior tendrían componentes estocásticos.

.

Apéndice A

Solución del nivel inferior problema binivel

Como se mencionó en el capítulo 3, para la solución del nivel inferior del problema binivel se utilizó el método de multiplicadores de Lagrange. La descripción de la metodología se presenta a continuación.

Recordemos que el problema del nivel inferior para cada almacén abierto $i, i \in I$ es:

$$\min_{Q_i, B_i} \sum_{j \in J} \left[\frac{A_i}{Q_i} y_{ij} d_j + \frac{x_i y_{ij} \pi_i B_i d_j}{Q_i} + \frac{x_i h_i (Q_i - B_i)^2}{2Q_i} + \frac{x_i \hat{\pi}_i (B_i)^2}{2Q_i} \right] \tag{A.1}$$

$$Q_i - B_i \leq I_i^{cap} x_i, \tag{A.2}$$

$$B_i x_i \leq \beta_i \tag{A.3}$$

$$Q_i > 0, \qquad B_i \geq 0 \tag{A.4}$$

sea:

$$f_i(Q_i, B_i) = \sum_{j \in J} \left[\frac{A_i}{Q_i} y_{ij} d_j + \frac{x_i y_{ij} \pi_i B_i d_j}{Q_i} + \frac{x_i h_i (Q_i - B_i)^2}{2Q_i} + \frac{x_i \hat{\pi}_i (B_i)^2}{2Q_i} \right] \tag{A.5}$$

$$g_1(Q_i, B_i) = Q_i - B_i - I_i^{cap} x_i, \tag{A.6}$$

$$g_2(Q_i, B_i) = B_i x_i - \beta_i, \tag{A.7}$$
$$g_3(Q_i, B_i) = -Q_i, \tag{A.8}$$
$$g_4(Q_i, B_i) = -B_i. \tag{A.9}$$

Utilizando la notación anterior, el problema del nivel inferior del almacén i, se reescribe como:

$$\begin{aligned} min \quad & f_i(Q_i, B_i) && \text{(A.10)}\\ s.a. \quad & g_1(Q_i, B_i) \leq 0 && \text{(A.11)}\\ & g_2(Q_i, B_i) \leq 0 && \text{(A.12)}\\ & g_3(Q_i, B_i) < 0 && \text{(A.13)}\\ & g_4(Q_i, B_i) \leq 0 && \text{(A.14)} \end{aligned}$$

Entonces podemos escribir la función de Lagrange:

$$L(Q_i, B_i, \lambda) = f_i(Q_i, B_i) + \lambda_1 g_1(Q_i, B_i)\lambda_2 g_2(Q_i, B_i) + \lambda_3 g_3(Q_i, B_i) + \lambda_4 g_4(Q_i, B_i)$$

Para el problema del nivel inferior del almacén i, el punto óptimo no puede satisfacer la restricción A.13 como igualdad, porque cuando $Q_i \to 0$ el valor de la función objetivo tiende a infinito. El óptimo tampoco puede ser un punto interior porque el gradiente de la región objetivo nunca vale cero en la región factible. Y además como la región factible es un polígono de cuatro lados, no se puede cumplir más de dos restricciones de igualdad al mismo tiempo.Entonces el óptimo debe caer en alguno de los siguientes casos:

- Caso 1.

  El punto óptimo solo cumple la restricción g_1 como igualdad, entonces, $\lambda_2 = \lambda_3 = \lambda_4 = 0$ y las condiciones de Lagrange quedan de la siguiente manera:

$$\frac{\partial L}{\partial Q_i} = \frac{\partial f_i(Q_i, B_i)}{\partial Q_i} + \lambda_1 \frac{\partial g_1(Q_i, B_i)}{\partial Q_i} = 0 \tag{A.15}$$
$$\frac{\partial L}{\partial B_i} = \frac{\partial f_i(Q_i, B_i)}{\partial B_i} + \lambda_1 \frac{\partial g_1(Q_i, B_i)}{\partial B_i} = 0 \tag{A.16}$$
$$\frac{\partial L}{\partial \lambda_1} = g_1(Q_i, B_i) = 0 \tag{A.17}$$

- Caso 2.

  El punto óptimo se cumple en g_1 y g_2. Entonces $\lambda_3 = \lambda_4 = 0$ Entonces, en este caso el óptimo es la intersección de las rectas:

$$g_1(Q_i, B_i) = 0 \tag{A.18}$$
$$g_2(Q_i, B_i) = 0 \tag{A.19}$$

- Caso 3.

  El punto óptimo se cumple en g_2. Entonces $\lambda_1 = \lambda_3 = \lambda_4 = 0$ y las condiciones de Lagrange quedan:

$$\frac{\partial L}{\partial Q_i} = \frac{\partial f_i(Q_i, B_i)}{\partial Q_i} + \lambda_2 \frac{\partial g_2(Q_i, B_i)}{\partial Q_i} = 0 \tag{A.20}$$
$$\frac{\partial L}{\partial B_i} = \frac{\partial f_i(Q_i, B_i)}{\partial B_i} + \lambda_2 \frac{\partial g_2(Q_i, B_i)}{\partial B_i} = 0 \tag{A.21}$$
$$\frac{\partial L}{\partial \lambda_2} = g_1(Q_i, B_i) = 0 \tag{A.22}$$

  Caso 4

  El punto óptimo se cumple en g_1 y g_4. Entonces $\lambda_1 = \lambda_2 = 0$ Entonces, en este caso el óptimo es la intersección de las rectas:

$$g_1(Q_i, B_i) = 0 \tag{A.23}$$
$$g_4(Q_i, B_i) = 0 \tag{A.24}$$

  Caso 5

  El punto óptimo se cumple en g_4. Entonces $\lambda_1 = \lambda_2 = \lambda_3 = 0$ y las condiciones de Lagrange quedan:

$$\frac{\partial L}{\partial Q_i} = \frac{\partial f_i(Q_i, B_i)}{\partial Q_i} + \lambda_4 \frac{\partial g_4(Q_i, B_i)}{\partial Q_i} = 0 \tag{A.25}$$
$$\frac{\partial L}{\partial B_i} = \frac{\partial f_i(Q_i, B_i)}{\partial B_i} + \lambda_4 \frac{\partial g_4(Q_i, B_i)}{\partial B_i} = 0 \tag{A.26}$$
$$\frac{\partial L}{\partial \lambda_4} = g_1(Q_i, B_i) = 0 \tag{A.27}$$

Una vez que se tenga la solución del sistema de ecuaciones de cada uno de los 5 casos (si existe), se evalúa en la función objetivo y la que tenga el menor valor, es considerado el mínimo global del problema del nivel inferior del almacén i (esta solución siempre existe).

Bibliografía

Abareshi, M. and Zaferanieh, M. (2019), 'A bi-level capacitated p-median facility location problem with the most likely allocation solution', *Transportation Research Part B: Methodological* **123**, 1–20.

Abo-Elnaga, Y., Nasr, S., El-Desoky, I., Hendawy, Z. and Mousa, A. (2021), A chaotic search-enhanced genetic algorithm for bilevel programming problems, *in* 'Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges', Springer, 129–146.

Acharya, N., Mahat, P. and Mithulananthan, N. (2006), 'An analytical approach for dg allocation in primary distribution network', *International Journal of Electrical Power & Energy Systems* **28**(10), 669–678.

Aggarwal, S. K., Saini, L. M. and Sood, V. (2021), 'Large wind farm layout optimization using nature inspired meta-heuristic algorithms', *IETE Journal of Research* 1–18.

Aghezzaf, E. (2005), 'Capacity planning and warehouse location in supply chains with uncertain demands', *Journal of the Operational Research Society* **56**(4), 453–462.

Ahmadi-Javid, A., Seyedi, P. and Syam, S. S. (2017), 'A survey of healthcare facility location', *Computers & Operations Research* **79**, 223–263.

Anton, M. (2020), Automated machine learning using evolutionary algorithms, *in* '2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)', IEEE, 101–107.

Aydilek, İ. B., Karaçizmeli, İ. H., Tenekeci, M. E., Kaya, S. and Gümüşçü, A. (2021), 'Using chaos enhanced hybrid firefly particle swarm optimization algorithm for solving continuous optimization problems', *Sādhanā* **46**(2), 1–22.

Barman, D. and Das, B. (2021), 'Two-echelon supply chain inventory model with variable lead time and ramp-type demand for deterioration item under bi-level credit period', *International Journal of Inventory Research* **6**(1), 14–46.

Battini, D., Persona, A. and Sgarbossa, F. (2014), 'A sustainable eoq model: Theoretical formulation and applications', *International Journal of Production Economics* **149**, 145–153.

Baumol, W. J. and Wolfe, P. (1958), 'A warehouse-location problem', *Operations research* **6**(2), 252–263.

Bautista, J. and Pereira, J. (2006), 'Modeling the problem of locating collection areas for urban waste management. an application to the metropolitan area of barcelona', *Omega* **34**(6), 617–629.

Boonmee, C., Arimura, M. and Asada, T. (2017), 'Facility location optimization model for emergency humanitarian logistics', *International Journal of Disaster Risk Reduction* **24**, 485–498.

Boyacı, B., Zografos, K. G. and Geroliminis, N. (2015), 'An optimization framework for the development of efficient one-way car-sharing systems', *European Journal of Operational Research* **240**(3), 718–733.

Bracken, J. and McGill, J. T. (1974), 'Defense applications of mathematical programs with optimization problems in the constraints', *Operations Research* **22**(5), 1086–1096.

Cabrera, G., Miranda, P. A., Cabrera, E., Soto, R., Crawford, B., Rubio, J. M. and Paredes, F. (2013), 'Solving a novel inventory location model with stochastic constraints and inventory control policy', *Mathematical Problems in Engineering* **2013**.

Calvete, H. I., Galé, C., Iranzo, J. A. and Toth, P. (2018), 'A matheuristic for the two-stage fixed-charge transportation problem', *Computers & Operations Research* **95**, 113–122.

Calvete, H. I., Galé, C. and Oliveros, M.-J. (2011), 'Bilevel model for production–distribution planning solved by using ant colony optimization', *Computers & operations research* **38**(1), 320–327.

Camacho-Vallejo, J.-F., Cordero-Franco, Á. E. and González-Ramírez, R. G. (2014), 'Solving the bilevel facility location problem under preferences by a stackelberg-evolutionary algorithm', *Mathematical Problems in Engineering* **2014**.

Chaiwuttisak, P., Smith, H., Wu, Y., Potts, C., Sakuldamrongpanich, T. and Pathomsiri, S. (2016), 'Location of low-cost blood collection and distribution centres in thailand', *Operations Research for Health Care* **9**, 7–15.

Chen, H., Wang, L., Di, J. and Ping, S. (2020), 'Bacterial foraging optimization based on self-adaptive chemotaxis strategy', *Computational intelligence and neuroscience* **2020**.

Chen, Y.-x., Tadikamalla, P. R., Shang, J. and Song, Y. (2020), 'Supply allocation: Bi-level programming and differential evolution algorithm for natural disaster relief', *Cluster Computing* **23**(1), 203–217.

Colson, B., Marcotte, P. and Savard, G. (2007), 'An overview of bilevel optimization', *Annals of operations research* **153**(1), 235–256.

Contardo, C., Iori, M. and Kramer, R. (2019), 'A scalable exact algorithm for the vertex p-center problem', *Computers & Operations Research* **103**, 211–220.

Cooper, L. (1963), 'Location-allocation problems', *Operations Research* **11**(3), 331–343.

CUI, G.-b. and LI, Y.-j. (2007), 'Study on the combined location routing and inventory problem in logistics system based on bi-level programming [j]', *Systems Engineering-Theory & Practice* **6**, 49–55.

Dai, W., Liang, L. and Zhang, B. (2020), 'Firefly optimization algorithm for the prediction of uplift due to high-pressure jet grouting', *Advances in Civil Engineering* **2020**.

Daskin, M. S. (2008), 'What you should know about location modeling', *Naval Research Logistics (NRL)* **55**(4), 283–294.

Daskin, M. S., Coullard, C. R. and Shen, Z.-J. M. (2002), 'An inventory-location model: Formulation, solution algorithm and computational results', *Annals of Operations Research* **110**(1-4), 83–106.

Del Prete, A., Franchi, R., Cacace, S. and Semeraro, Q. (2020), 'Optimization of cutting conditions using an evolutive online procedure', *Journal of Intelligent Manufacturing* **31**(2), 481–499.

Dempe, S. (2002), *Foundations of bilevel programming*, Springer Science and Business Media.

Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L. and Shankar, K. (2020), 'Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments', *Journal of Parallel and Distributed Computing* **142**, 36–45.

Diabat, A., Abdallah, T. and Henschel, A. (2015), 'A closed-loop location-inventory problem with spare parts consideration', *Computers & Operations Research* **54**, 245–256.

Diabat, A. and Theodorou, E. (2015), 'A location–inventory supply chain problem: Reformulation and piecewise linearization', *Computers & Industrial Engineering* **90**, 381–389.

Du, B., Zhou, H. and Leus, R. (2020), 'A two-stage robust model for a reliable p-center facility location problem', *Applied Mathematical Modelling* **77**, 99–114.

Emeç, Ş. and Akkaya, G. (2018), 'Stochastic ahp and fuzzy vikor approach for warehouse location selection problem', *Journal of Enterprise Information Management* .

Farahani, R. Z., Hekmatfar, M., Fahimnia, B. and Kazemzadeh, N. (2014), 'Hierarchical facility location problem: Models, classifications, techniques, and applications', *Computers & Industrial Engineering* **68**, 104–117.

Farshi, T. R. and Orujpour, M. (2021), 'A multi-modal bacterial foraging optimization algorithm', *Journal of Ambient Intelligence and Humanized Computing* 1–15.

Gebicki, M., Mooney, E., Chen, S.-J. G. and Mazur, L. M. (2014), 'Evaluation of hospital medication inventory policies', *Health care management science* **17**(3), 215–229.

Ghasemi, P. and Khalili-Damghani, K. (2021), 'A robust simulation-optimization approach for pre-disaster multi-period location–allocation–inventory planning', *Mathematics and computers in simulation* **179**, 69–95.

Ghodratnama, A., Arbabi, H. and Azaron, A. (2019), 'Production planning in industrial townships modeled as hub location–allocation problems considering congestion in manufacturing plants', *Computers & Industrial Engineering* **129**, 479–501.

Godini, K., Azarian, G., Kimiaei, A., Dragoi, E. N. and Curteanu, S. (2021), 'Modeling of a real industrial wastewater treatment plant based on aerated lagoon using a neuro-evolutive technique', *Process Safety and Environmental Protection* **148**, 114–124.

Goldberg, D. (1989), Genetic algorithms in search, optimization and machine learning., *in* 'Addison-Wesley Longman Publishing Co., Inc. Boston'.

Govindan, K., Jafarian, A., Khodaverdi, R. and Devika, K. (2014), 'Two-echelon multiple-vehicle location–routing problem with time windows for optimization of sustainable supply chain network of perishable food', *International Journal of Production Economics* **152**, 9–28.

Guan, R. and Zhao, X. (2011), 'Pricing and inventory management in a system with multiple competing retailers under (r, q) policies', *Computers & Operations Research* **38**(9), 1294–1304.

Gümüs, A. T. and Güneri, A. F. (2007), 'Multi-echelon inventory management in supply chains with uncertain demand and lead times: literature review from an operational research perspective', *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **221**(10), 1553–1570.

Hamdia, K. M., Zhuang, X. and Rabczuk, T. (2021), 'An efficient optimization approach for designing machine learning models based on genetic algorithm', *Neural Computing and Applications* **33**(6), 1923–1933.

Hanjoul, P. and Peeters, D. (1987), 'A facility location problem with clients' preference orderings', *Regional Science and Urban Economics* **17**(3), 451–473.

Hansen, P., Jaumard, B. and Savard, G. (1992), 'New branch-and-bound rules for linear bilevel programming', *SIAM Journal on scientific and Statistical Computing* **13**(5), 1194–1217.

Izdebski, M., Jacyna-Gołda, I., Wasiak, M., Jachimowski, R., Kłodawski, M., Pyza, D. and Żak, J. (2018), 'The application of the genetic algorithm to multi-criteria warehouses location problems on the logistics network', *Transport* **33**(3), 741–750.

Jeroslow, R. G. (1985), 'The polynomial hierarchy and a simple model for competitive analysis', *Mathematical programming* **32**(2), 146–164.

Jia, H., Ordonez, F. and Dessouky, M. M. (2007), 'Solution approaches for facility location of medical supplies for large-scale emergencies', *Computers & Industrial Engineering* **52**(2), 257–276.

Jiao, Y.-l., Xing, X.-c., Zhang, P., Xu, L.-c. and Liu, X.-R. (2018), 'Multi-objective storage location allocation optimization and simulation analysis of automated warehouse based on multi-population genetic algorithm', *Concurrent Engineering* **26**(4), 367–377.

Kalashnikov, V. V., Dempe, S., Perez-Valdes, G. A., Kalashnykova, N. I. and Camacho-Vallejo, J.-F. (2015), 'Bilevel programming and applications', *Mathematical Problems in Engineering* **2015**.

Katoch, S., Chauhan, S. S. and Kumar, V. (2021), 'A review on genetic algorithm: past, present, and future', *Multimedia Tools and Applications* **80**(5), 8091–8126.

Kim, C., Batra, R., Chen, L., Tran, H. and Ramprasad, R. (2021), 'Polymer design using genetic algorithm and machine learning', *Computational Materials Science* **186**, 110067.

Kongsomsaksakul, S., Yang, C. and Chen, A. (2005), 'Shelter location-allocation model for flood evacuation planning', *Journal of the Eastern Asia Society for Transportation Studies* **6**, 4237–4252.

Kusmindarti, D. (2009), 'Penentuan lokasi distributor dengan menggunakan analytical hierarchy process dan skala liberatore', *Jurnal Teknik Industri* **10**(1), 13–19.

Lien, J.-M., Rodriguez, S. and Morales, M. (2021), 'Persistent covering with latency and energy constraints', *IEEE Robotics and Automation Letters* **6**(2), 998–1003.

Lin, J.-R. and Yang, T.-H. (2011), 'Strategic design of public bicycle sharing systems with service level constraints', *Transportation research part E: logistics and transportation review* **47**(2), 284–294.

Lloyd, S. (1957), 'Least square quantization in pcm. bell telephone laboratories paper. published in journal much later: Lloyd, sp: Least squares quantization in pcm', *IEEE Trans. Inform. Theor.(1957/1982)* **18**.

Lu, Z. and Bostel, N. (2007), 'A facility location model for logistics systems including reverse flows: The case of remanufacturing activities', *Computers & Operations Research* **34**(2), 299–323.

Luo, L., Zhang, Z. and Yin, Y. (2021), 'Simulated annealing and genetic algorithm based method for a bi-level seru loading problem with worker assignment in seru production systems', *Journal of Industrial & Management Optimization* **17**(2), 779.

Ma, Y., Xu, W., Qin, L., Zhao, X. and Du, J. (2019), 'Hierarchical supplement location-allocation optimization for disaster supply warehouses in the beijing–tianjin–hebei region of china', *Geomatics, Natural Hazards and Risk* **10**(1), 102–117.

Mahmoud, N. M., Fouad, H., Alsadon, O. and Soliman, A. M. (2020), 'Detecting dental problem related brain disease using intelligent bacterial optimized associative deep neural network', *Cluster Computing* **23**(3), 1647–1657.

Mahmud, A. R. and Indriasari, V. (2009), 'Facility location models development to maximize total service area', *Theoretical and Empirical Researches in Urban Management* **4**(1S), 87–100.

Marinakis, Y. and Marinaki, M. (2008), 'A bilevel genetic algorithm for a real life location routing problem', *International Journal of Logistics: Research and Applications* **11**(1), 49–65.

Masudin, I. (2013), 'Facility location modeling in multi-echelon distribution system: A case study of indonesian liquefied petroleum gas supply chain', *Aceh International Journal of Science and Technology* **2**(1), 37–43.

Masudin, I. (2015), 'An investigation of the relationship between facility location decisions, service level and distribution costs: A proposed model for indonesian lpg supply chain', *International Journal of Business and Society (IJBS)* **16**(1), 117–132.

Matheus, P. and Gelders, L. (2000), 'The (r, q) inventory policy subject to a compound poisson demand pattern', *International Journal of Production Economics* **68**(3), 307–317.

Mirjalili, S. (2019), Genetic algorithm, *in* 'Evolutionary algorithms and neural networks', Springer, 43–55.

Mousavi, S. M., Alikar, N., Niaki, S. T. A. and Bahreininejad, A. (2015), 'Optimizing a location allocation-inventory problem in a two-echelon supply chain network: A modified fruit fly optimization algorithm', *Computers & Industrial Engineering* **87**, 543–560.

Muni, M. K., Parhi, D. R. and Kumar, P. B. (2021), 'Improved motion planning of humanoid robots using bacterial foraging optimization', *Robotica* **39**(1), 123–136.

Murray, A. T. and Gerrard, R. A. (1997), 'Capacitated service and regional constraints in location-allocation modeling', *Location science* **5**(2), 103–118.

Nozick, L. K. and Turnquist, M. A. (1998), 'Integrating inventory impacts into a fixed-charge model for locating distribution centers', *Transportation Research Part E: Logistics and Transportation Review* **34**(3), 173–186.

Nucamendi-Guillén, S., Dávila, D., Camacho-Vallejo, J.-F. and González-Ramírez, R. G. (2018), 'A discrete bilevel brain storm algorithm for solving a sales territory design problem: a case study', *Memetic Computing* **10**(4), 441–458.

Osborn, A. F. (1957), 'Applied imagination, revised ed', *New York: Scribner* .

Özceylan, E., Uslu, A., Erbaş, M., Çetinkaya, C. and İşleyen, S. K. (2017), 'Optimizing the location-allocation problem of pharmacy warehouses: A case study in gaziantep', *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)* **7**(1), 117–129.

Ozsen, L., Coullard, C. R. and Daskin, M. S. (2008), 'Capacitated warehouse location model with risk pooling', *Naval Research Logistics (NRL)* **55**(4), 295–312.

Paniri, M., Dowlatshahi, M. B. and Nezamabadi-pour, H. (2020), 'Mlaco: A multi-label feature selection algorithm based on ant colony optimization', *Knowledge-Based Systems* **192**, 105285.

Perl, J. and Sirisoponsilp, S. (1988), 'Distribution networks: facility location, transportation and inventory', *International Journal of Physical Distribution & Materials Management* .

Piotrowski, A. P., Napiorkowski, J. J. and Piotrowska, A. E. (2020), 'Population size in particle swarm optimization', *Swarm and Evolutionary Computation* **58**, 100718.

Pop, P. C., Sabo, C., Biesinger, B., Hu, B. and Raidl, G. R. (2017), 'Solving the two-stage fixed-charge transportation problem with a hybrid genetic algorithm', *Carpathian Journal of Mathematics* **33**(3), 365–371.

Rahman, S.-u. and Smith, D. K. (2000), 'Use of location-allocation models in health service development planning in developing nations', *European Journal of Operational Research* **123**(3), 437–452.

Randhawa, S. U. and West, T. M. (1995), 'An integrated approach to facility location problems', *Computers & Industrial Engineering* **29**(1-4), 261–265.

Saeedi Mehrabad, M., Aazami, A. and Goli, A. (2017), 'A location-allocation model in the multi-level supply chain with multi-objective evolutionary approach', *Journal of Industrial and Systems Engineering* **10**(3), 140–160.

Şahin, G., Süral, H. and Meral, S. (2007), 'Locational analysis for regionalization of turkish red crescent blood services', *Computers & Operations Research* **34**(3), 692–704.

Sayah, D. and Irnich, S. (2017), 'A new compact formulation for the discrete p-dispersion problem', *European Journal of Operational Research* **256**(1), 62–67.

Schuijbroek, J., Hampshire, R. C. and Van Hoeve, W.-J. (2017), 'Inventory rebalancing and vehicle routing in bike sharing systems', *European Journal of Operational Research* **257**(3), 992–1004.

Sennan, S., Somula, R., Luhach, A. K., Deverajan, G. G., Alnumay, W., Jhanjhi, N., Ghosh, U. and Sharma, P. (2020), 'Energy efficient optimal parent selection based routing protocol for internet of things using firefly optimization algorithm', *Transactions on Emerging Telecommunications Technologies* e4171.

Shen, Z.-J. M., Coullard, C. and Daskin, M. S. (2003), 'A joint location-inventory model', *Transportation science* **37**(1), 40–55.

Shen, Z.-J. M. and Qi, L. (2007), 'Incorporating inventory and routing costs in strategic location models', *European journal of operational research* **179**(2), 372–389.

Sherbrooke, C. C. (1968), 'Metric: A multi-echelon technique for recoverable item control', *Operations research* **16**(1), 122–141.

Shi, Y. (2011), Brain storm optimization algorithm, *in* 'International Conference in Swarm Intelligence', Springer, 303–309.

Suzuki, A. and Drezner, Z. (1996), 'The p-center location problem in an area', *Location science* **4**(1-2), 69–82.

Temur, G. T., Kaya, T. and Kahraman, C. (2014), Facility location selection in reverse logistics using a type-2 fuzzy decision aid method, *in* 'Supply chain management under fuzziness', Springer, 591–606.

Tutunchi, G. K. and Fathi, Y. (2019), 'Effective methods for solving the bi-criteria p-center and p-dispersion problem', *Computers & Operations Research* **101**, 43–54.

Uribe, H. A., Torres, L. G., Félix, I. Z. and Hernández, Z. S. (2019), 'Diseño e implementación de un algoritmo genético para la predicción de una variable.', *Res. Comput. Sci.* **148**(8), 187–197.

Usman, Y. V., Ismail, A. H., Hidayah, N. Y. and Chairani, L. (2013), 'Pengembangan model pemilihan lokasi pembuangan akhir sampah perkotaan (studi kasus: Kota jakarta timur)'.

Vicente, L. N. and Calamai, P. H. (1994), 'Bilevel and multilevel programming: A bibliography review', *Journal of Global optimization* **5**(3), 291–306.

Wang, F., Zhang, H. and Zhou, A. (2021), 'A particle swarm optimization algorithm for mixed-variable optimization problems', *Swarm and Evolutionary Computation* **60**, 100808.

Wang, Y., Pan, S., Al-Shihabi, S., Zhou, J., Yang, N. and Yin, M. (2021), 'An improved configuration checking-based algorithm for the unicost set covering problem', *European Journal of Operational Research* .

Wang, Z.-J., Zhan, Z.-H., Kwong, S., Jin, H. and Zhang, J. (2020), 'Adaptive granularity learning distributed particle swarm optimization for large-scale optimization', *IEEE transactions on cybernetics* **51**(3), 1175–1188.

Wang, Z., Yao, D.-Q. and Huang, P. (2007), 'A new location-inventory policy with reverse logistics applied to b2c e-markets of china', *International Journal of Production Economics* **107**(2), 350–363.

Wati, P. E. D. K. and Nuha, H. (2018), 'Pengembangan model capacitated maximal covering location problem (cmclp) dalam penentuan lokasi pendirian gudang', *Jurnal Teknik Industri* **19**(1), 21–27.

Wen, U. P. and Hsu, S.-T. (1991), 'Linear bi level programming problems a review', *Journal of the Operational Research Society* **42**(2), 125–133.

Wu, Q., Du, Z., Zhao, Y., Xu, H. and Zhang, X. (2021), 'Optimal location of water level sensors for monitoring mine water inrush based on the set covering model', *Scientific Reports* **11**(1), 1–12.

Xu, P. (1999), 'Solving a joint inventory-location problem by a substitution algorithm', *Undergraduate senior thesis, Northwestern University* .

Yang, K., You, X., Liu, S. and Pan, H. (2020), 'A novel ant colony optimization based on game for traveling salesman problem', *Applied Intelligence* **50**(12), 4529–4542.

Yao, Z., Lee, L. H., Jaruphongsa, W., Tan, V. and Hui, C. F. (2010), 'Multi-source facility location–allocation and inventory problem', *European Journal of Operational Research* **207**(2), 750–762.

Yuan, G., Gao, Y., Ye, B. and Liu, Z. (2021), 'A bilevel programming approach for real-time pricing strategy of smart grid considering multi-microgrids connection', *International Journal of Energy Research* .

Zhang, Y., Liu, X., Bao, F., Chi, J., Zhang, C. and Liu, P. (2020), 'Particle swarm optimization with adaptive learning strategy', *Knowledge-Based Systems* **196**, 105789.

Zhao, D., Liu, L., Yu, F., Heidari, A. A., Wang, M., Oliva, D., Muhammad, K. and Chen, H. (2021), 'Ant colony optimization with horizontal and vertical crossover search: Fundamental visions for multi-threshold image segmentation', *Expert Systems with Applications* **167**, 114122.

Zhu, S., Hu, X., Huang, K. and Yuan, Y. (2021), 'Optimization of product category allocation in multiple warehouses to minimize splitting of online supermarket customer orders', *European Journal of Operational Research* **290**(2), 556–571.